OpenNebula.org

OpenNebula 4.4 Integration Guide

OpenNebula Project

February 05, 2014

CONTENTS

CHAPTER

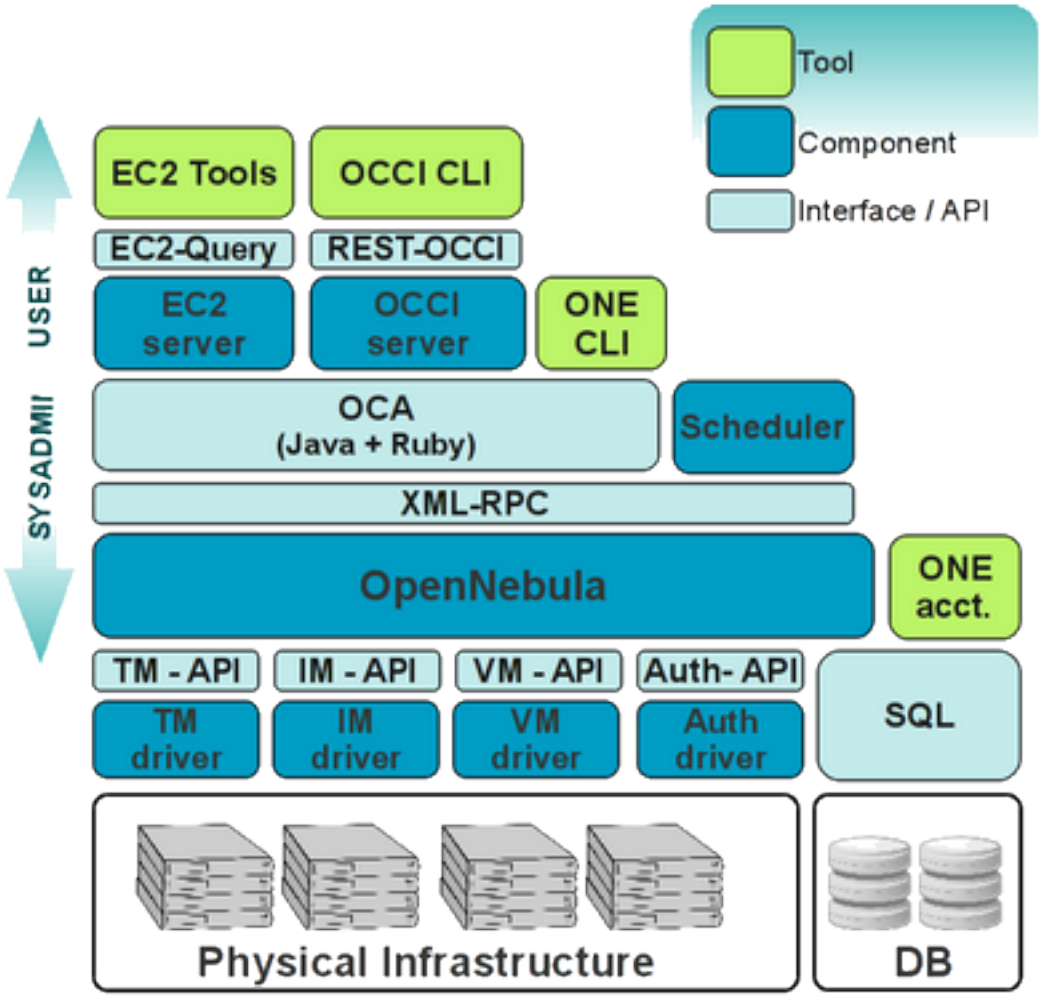
ONE

GETTING STARTED

1.1 Scalable Architecture and APIs

OpenNebula has been designed to be easily adapted to any infrastructure and easily extended with new components. The result is a modular system that can implement a variety of Cloud architectures and can interface with multiple datacenter services. In this Guide we review the main interfaces of OpenNebula, their use and give pointers to additional documentation for each one.

We have classified the interfaces in two categories: end-user cloud and system interfaces. Cloud interfaces are primary used to develop tools targeted to the end-user, and they provide a high level abstraction of the functionality provided by the Cloud. On the other hand, the system interfaces expose the full functionality of OpenNebula and are mainly used to adapt and tune the behavior of OpenNebula to the target infrastructure.



1.1.1 1. Cloud Interfaces

Cloud interfaces enable you to manage virtual machines, networks and images through a simple and easy-to-use REST API. The Cloud interfaces hide most of the complexity of a Cloud and are specially suited for end-users. OpenNebula implements two different interfaces, namely:

- **EC2-Query API**. OpenNebula implements the functionality offered by the Amazon's EC2 API, mainly those related to virtual machine management. In this way, you can use any EC2 Query tool to access your OpenNebula Cloud.
- **OCCI-OGF**. The OpenNebula OCCI API is a RESTful service to create, control and monitor cloud resources using an implementation of the OGF OCCI API specification based on the draft 0.8

Use the cloud interface if... you are developing portals, tools or specialized solutions for end-users.

You can find more information at... *EC2-Query reference*, and *OCCI reference guides*.

1.1.2 2. System Interfaces

2.1. The OpenNebula XML-RPC Interface

The XML-RPC interface is the primary interface for OpenNebula, and it exposes all the functionality to interface the OpenNebula daemon. Through the XML-RPC interface you can control and manage any OpenNebula resource, including virtual machines, networks, images, users, hosts and clusters.

Use the XML-RPC interface if... you are developing specialized libraries for Cloud applications or you need a low-level interface with the OpenNebula core.

You can find more information at... *XML-RPC reference guide*.

2.2. The OpenNebula Cloud API (OCA)

The OpenNebula Cloud API provides a simplified and convenient way to interface the OpenNebula core. The OCA interfaces exposes the same functionality as that of the XML-RPC interface. OpenNebula includes two language bindings for OCA: Ruby and JAVA.

Use the OCA interface if... you are developing advanced IaaS tools that need full access to the OpenNebula functionality.

You can find more information at... *OCA-Ruby reference guide* and the *OCA-JAVA reference guide*.

2.3. The OpenNebula Drivers Interfaces

The interactions between OpenNebula and the Cloud infrastructure are performed by specific drivers each one addressing a particular area:

- **Storage**. The OpenNebula core issue abstract storage operations (e.g. clone or delete) that are implemented by specific programs that can be replaced or modified to interface special storage backends and file-systems.
- **Virtualization**. The interaction with the hypervisors are also implemented with custom programs to boot, stop or migrate a virtual machine. This allows you to specialize each VM operation so to perform custom operations.
- **Monitoring**. Monitoring information is also gathered by external probes. You can add additional probes to include custom monitoring metrics that can be later used to allocate virtual machines or for accounting purposes
- **Authorization**. OpenNebula can be also configured to use an external program to authorize and authenticate user requests. In this way, you can implement any access policy to Cloud resources.
- **Networking** the hypervisor is also prepared with the network configuration for each Virtual Machine.

Use the driver interfaces if... you need OpenNebula to interface any specific storage, virtualization, monitoring or authorization system already deployed in your datacenter or to tune the behavior of the standard OpenNebula drivers.

You can find more information at... the *virtualization system*,*storage system*, the *information system*, the *authentication system* and *network system* guides.

2.4. The OpenNebula DataBase

OpenNebula saves its state and lots of accounting information in a persistent data-base. OpenNebula can use MySQL or SQLite database that can be easily interfaced with any of DB tool.

Use the OpenNebula DB if... you need to generate custom accounting or billing reports.

CHAPTER

TWO

CLOUD INTERFACES

2.1 OpenNebula OCCI Specification

2.1.1 Overview

The OpenNebula OCCI API is a RESTful service to create, control and monitor cloud resources using an implementation of the OGF OCCI API specification based on the draft 0.8. This implementation also includes some extensions, requested by the community, to support OpenNebula specific functionality. There are two types of resources that resemble the basic entities managed by the OpenNebula system, namely:

- **Pool Resources (PR)**: Represents a collection of elements owned by a given user. In particular five collections are defined:

```
<COLLECTIONS>
  <COMPUTE_COLLECTION href="http://localhost:4567/compute">
  <INSTANCE_TYPE_COLLECTION href="http://localhost:4567/instance_type">
  <NETWORK_COLLECTION href="http://localhost:4567/network">
  <STORAGE_COLLECTION href="http://localhost:4567/storage">
  <USER_COLLECTION href="http://localhost:4567/user">
</COLLECTIONS>
```

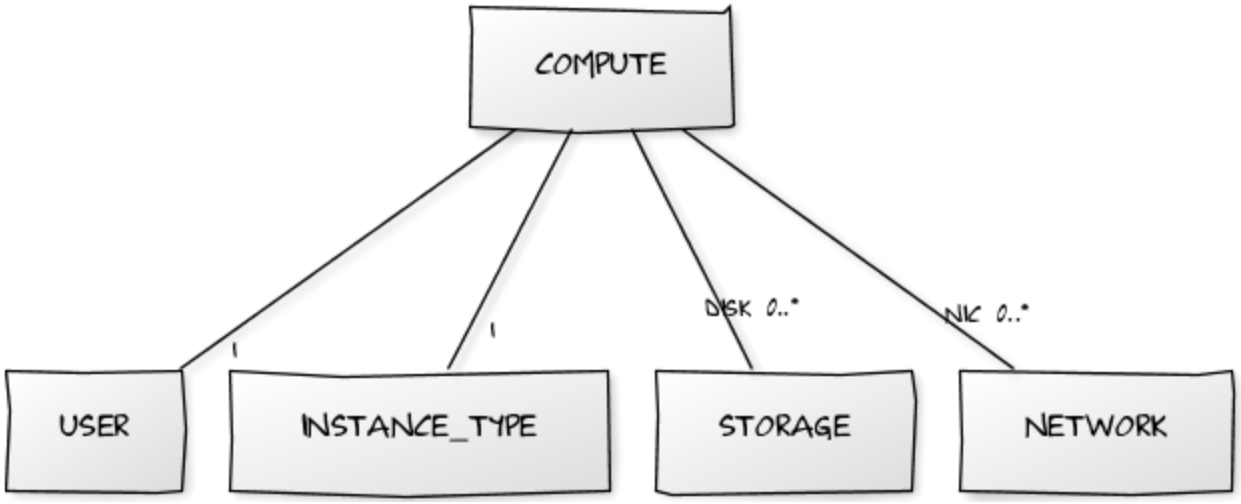
- **Entry Resources (ER)**: Represents a single entry within a given collection: COMPUTE, NETWORK, STORAGE, INSTANCE_TYPE and USER.

Each one of ERs in the pool are described by an element (e.g. `COMPUTE`, `INSTANCE_TYPE`, `NETWORK`, `STORAGE` or `USER`) with one attribute:

- `href`, a URI for the ER

```
<COMPUTE_COLLECTION>
    <COMPUTE href="http://www.opennebula.org/compute/310" name="TestVM"/>
    <COMPUTE href="http://www.opennebula.org/compute/432" name="Server1"/>
    <COMPUTE href="http://www.opennebula.org/compute/123" name="Server2"/>
</COMPUTE_COLLECTION>
```

A `COMPUTE` entry resource can be linked to one or more `STORAGE` or `NETWORK` resources and one `INSTANCE_TYPE` and `USER`.



2.1.2 Authentication & Authorization

User authentication will be HTTP Basic access authentication to comply with REST philosophy. The credentials passed should be the User name and password. If you are not using the occi tools provided by OpenNebula, the password has to be SHA1 hashed as well as it is stored in the database instead of using the plain version.

2.1.3 HTTP Headers

The following headers are compulsory:

- **Content-Length**: The size of the Entity Body in octets
- **Content-Type**: application/xml

Uploading images needs HTTP multi part support, and also the following header

- **Content-Type**: multipart/form-data

2.1.4 Return Codes

The OpenNebula Cloud API uses the following subset of HTTP Status codes:

- **200 OK** : The request has succeeded.
- **201 Created** : Request was successful and a new resource has being created
- **202 Accepted** : The request has been accepted for processing, but the processing has not been completed
- **204 No Content** : The request has been accepted for processing, but no info in the response
- **400 Bad Request** : Malformed syntax
- **401 Unauthorized** : Bad authentication
- **403 Forbidden** : Bad authorization
- **404 Not Found** : Resource not found
- **500 Internal Server Error** : The server encountered an unexpected condition which prevented it from fulfilling the request.

- **501 Not Implemented** : The functionality requested is not supported

The methods specified below are described without taking into account **4xx** (can be inferred from authorization information in section above) and **5xx** errors (which are method independent). HTTP verbs not defined for a particular entity will return a **501 Not Implemented**.

2.1.5 Resource Representation

Network

The NETWORK element defines a virtual network that interconnects those COMPUTES with a network interface card attached to that network. The traffic of each network is isolated from any other network, so it constitutes a broadcasting domain.

The following elements define a NETWORK:

- ID, the uuid of the NETWORK
- NAME describing the NETWORK
- USER link to the USER owner of the NETWORK
- GROUP of the NETWORK
- DESCRIPTION of the NETWORK
- ADDRESS, of the NETWORK
- SIZE, of the network, defaults to C

The elements in bold can be provided in a POST request in order to create a new NETWORK resource based on those parameters.

Example:

```
<NETWORK href="http://www.opennebula.org/network/123">
     <ID>123</ID>
     <NAME>BlueNetwork</NAME>
     <USER href="http://www.opennebula.org/user/33" name="cloud_user"/>
     <GROUP>cloud_group</GROUP>
     <DESCRIPTION>This NETWORK is blue<DESCRIPTION>
     <ADDRESS>192.168.0.1</ADDRESS>
     <SIZE>C</SIZE>
</NETWORK>
```

Storage

The STORAGE is a resource containing an operative system or data, to be used as a virtual machine disk:

- ID the uuid of the STORAGE
- NAME describing the STORAGE
- USER link to the USER owner of the STORAGE
- GROUP of the STORAGE
- DESCRIPTION of the STORAGE
- TYPE, type of the image
  - OS: contains a working operative system

  - CDROM: readonly data
  - DATABLOCK: storage for data, which can be accessed and modified from different Computes
- SIZE, of the image in MBs
- FSTYPE, in case of DATABLOCK, the type of filesystem desired

The elements in bold can be provided in a POST request in order to create a new NETWORK resource based on those parameters.

Example:

```
<STORAGE href="http://www.opennebula.org/storage/123">
    <ID>123</ID>
    <NAME>Ubuntu Desktop</NAME>
    <USER href="http://www.opennebula.org/user/33" name="cloud_user"/>
    <GROUP>cloud_group</GROUP>
    <DESCRIPTION>Ubuntu 10.04 desktop for students.</DESCRIPTION>
    <TYPE>OS</TYPE>
    <SIZE>2048</SIZE>
</STORAGE>
```

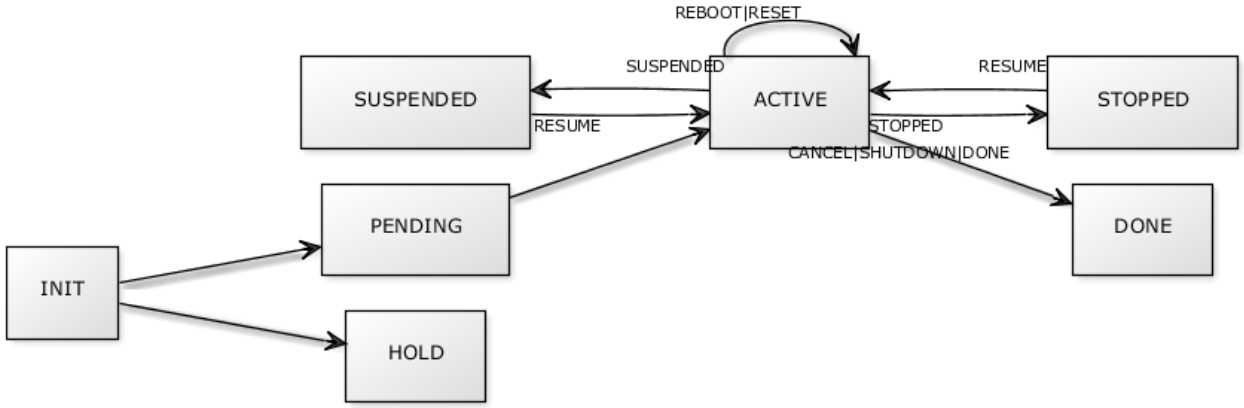
Compute

The COMPUTE element defines a virtual machine by specifying its basic configuration attributes such as NIC or DISK.

The following elements define a COMPUTE:

- ID, the uuid of the COMPUTE.
- NAME, describing the COMPUTE.
- USER link to the USER owner of the COMPUTE
- GROUP of the COMPUTE
- CPU number of CPUs of the COMPUTE
- MEMORY MBs of MEMORY of the COMPUTE
- INSTANCE_TYPE, ink to a INSTANCE_TYPE resource
- DISK, the block devices attached to the virtual machine.
  - STORAGE link to a STORAGE resource
  - TARGET
  - SAVE_AS link to a STORAGE resource to save the disk image when the COMPUTE is DONE
  - TYPE
- NIC, the network interfaces.
  - NETWORK link to a NETWORK resource
  - IP
  - MAC
- CONTEXT, key value pairs to be passed on creation to the COMPUTE.
  - KEY1 VALUE1
  - KEY2 VALUE2

- `STATE`, the state of the COMPUTE. This can be one of:



Example:

```
<COMPUTE href="http://www.opennebula.org/compute/32">
    <ID>32</ID>
    <NAME>Web Server</NAME>
    <CPU>1</CPU>
    <MEMORY>1024</MEMORY>
    <USER href="http://0.0.0.0:4567/user/310" name="cloud_user"/>
    <GROUP>cloud_group</GROUP>
    <INSTANCE_TYPE href="http://0.0.0.0:4567/instance_type/small">small</INSTANCE_TYPE>
    <STATE>ACTIVE</STATE>
    <DISK>
        <STORAGE href="http://www.opennebula.org/storage/34" name="Ubuntu10.04"/>
        <TYPE>OS</TYPE>
        <TARGET>hda</TARGET>
    </DISK>
    <DISK>
        <STORAGE href="http://www.opennebula.org/storage/24" name="testingDB"/>
        <SAVE_AS href="http://www.opennebula.org/storage/54"/>
        <TYPE>CDROM</TYPE>
        <TARGET>hdc</TARGET>
    </DISK>
    <NIC>
        <NETWORK href="http://www.opennebula.org/network/12" name="Private_LAN"/>
        <MAC>00:ff:72:31:23:17</MAC>
        <IP>192.168.0.12</IP>
    </NIC>
    <NIC>
        <NETWORK href="http://www.opennebula.org/network/10" name="Public_IPs"/>
        <MAC>00:ff:72:17:20:27</MAC>
        <IP>192.168.0.25</IP>
    </NIC>
    <CONTEXT>
        <PUB_KEY>FDASF324DSFA3241DASF</PUB_KEY>
    </CONTEXT>
</COMPUTE>
```

Instance type

An INSTANCE_TYPE specifies the COMPUTE capacity values

- `ID`, the uuid of the INSTANCE_TYPE.

- NAME, describing the INSTANCE_TYPE.
- CPU number of CPUs of the INSTANCE_TYPE
- MEMORY MBs of MEMORY of the INSTANCE_TYPE

Example:

```
<INSTANCE_TYPE href="http://www.opennebula.org/instance_type/small">
    <ID>small</ID>
    <NAME>small</NAME>
    <CPU>1</CPU>
    <MEMORY>1024</MEMORY>
</INSTANCE_TYPE>
```

User

A USER specifies the COMPUTE capacity values

- ID, the uuid of the INSTANCE_TYPE.
- NAME, describing the INSTANCE_TYPE.
- GROUP, fo the USER
- QUOTA,
  - CPU:
  - MEMORY:
  - NUM_VMS:
  - STORAGE
- USAGE,
  - CPU:
  - MEMORY:
  - NUM_VMS:
  - STORAGE

Example:

```
<USER href="http://www.opennebula.org/user/42">
    <ID>42</ID>
    <NAME>cloud_user</NAME>
    <GROUP>cloud_group</GROUP>
    <QUOTA>
        <CPU>8</CPU>
        <MEMORY>4096</MEMORY>
        <NUM_VMS>10</NUM_VMS>
        <STORAGE>0</STORAGE>
    </QUOTA>
    <USAGE>
        <CPU>2</CPU>
        <MEMORY>512</MEMORY>
        <NUM_VMS>2</NUM_VMS>
        <STORAGE>0</STORAGE>
    </USAGE>
</USER>
```

2.1.6 Request Methods

Method	URL	Meaning / Entity Body	Response
GET	/	**List** the available collections in the cloud.	**200 OK**: An XML representation of the the available collections in the http body

Network

Method	URL	Meaning / Entity Body	Response
GET	/network	**List** the contents of the NETWORK collection. Optionally a verbose param (/network?verbose=true) can be provided to retrieve an extended version of the collection	**200 OK**: An XML representation of the collection in the http body
POST	/network	**Create** a new NETWORK. An XML representation of a NETWORK without the ID element should be passed in the http body	**201 Created**: An XML representation of the new NETWORK with the ID
GET	/network/<id>	**Show** the NETWORK resource identified by <id>	**200 OK** : An XML representation of the NETWORK in the http body
PUT	/network/<id>	**Update** the NETWORK resource identified by <id>	**202 Accepted** : The update request is being process, polling required to confirm update
DELETE	/network/<id>	**Delete** the NETWORK resource identified by <id>	**204 No Content**:

Storage

Method	URL	Meaning / Entity Body	Response
GET	/storage	**List** the contents of the STORAGE collection. Optionally a verbose param (/storage?verbose=true) can be provided to retrieve an extended version of the collection	**200 OK**: An XML representation of the collection in the http body
POST	/storage	**Create** an new STORAGE. An XML representation of a STORAGE without the ID element should be passed in the http body	**201 Created**: An XML representation of the new NETWORK with the ID
GET	/storage/<id>	**Show** the STORAGE resource identified by <id>	**200 OK** : An XML representation of the STORAGE in the http body
PUT	/storage/<id>	**Update** the STORAGE resource identified by <id>	**202 Accepted** : The update request is being process, polling required to confirm update
DELETE	/storage/<id>	**Delete** the STORAGE resource identified by <id>	**204 No Content**:

Compute

Method	URL	Meaning / Entity Body	Response
GET	`/compute`	**List** the contents of the COMPUTE collection. Optionally a verbose param (`/compute?verbose=true`) can be provided to retrieve an extended version of the collection	**200 OK**: An XML representation of the pool in the http body
POST	`/compute`	**Create** a new COMPUTE. An XML representation of a COMPUTE without the ID element should be passed in the http body	**201 Created**: An XML representation of the new COMPUTE with the ID
GET	`/compute/<id>`	**Show** the COMPUTE resource identified by <id>	**200 OK** : An XML representation of the network in the http body
PUT	`/compute/<id>`	**Update** the COMMPUTE resource identified by <id>	**202 Accepted** : The update request is being process, polling required to confirm update
DELETE	`/compute/<id>`	**Delete** the COMPUTE resource identified by <id>	**204 No Content**: The Network has been successfully deleted

Instance type

Method	URL	Meaning / Entity Body	Response
GET	`/instance_type`	**List** the contents of the INSTANCE_TYPE collection. Optionally a verbose param (`/instance_type?verbose=true`) can be provided to retrieve an extended version of the collection	**200 OK**: An XML representation of the collection in the http body
GET	`/instance_type/<id>`	**Show** the INSTANCE_TYPE resource identified by <id>	**200 OK** : An XML representation of the INSTANCE_TYPE in the http body

User

Method	URL	Meaning / Entity Body	Response
GET	`/user`	**List** the contents of the USER collection. Optionally a verbose param (`/user?verbose=true`) can be provided to retrieve an extended version of the collection	**200 OK**: An XML representation of the collection in the http body
GET	`/user/<id>`	**Show** the USER resource identified by <id>	**200 OK** : An XML representation of the USER in the http body

2.1.7 Implementation Notes

Authentication

It is recommended that the server-client communication is performed over HTTPS to avoid sending user authentication information in plain text.

Notifications

HTTP protocol does not provide means for notification, so this API relies on asynchronous polling to find whether a RESOURCE update is successful or not.

2.1.8 Examples

CHAPTER

THREE

SYSTEM INTERFACES

3.1 XML-RPC API

This reference documentation describes the xml-rpc methods exposed by OpenNebula. Each description consists of the method name and the input and output values.

All xml-rpc responses share a common structure.

Type	Data Type	Description
OUT	Boolean	True or false whenever is successful or not.
OUT	String	If an error occurs this is the error message.
OUT	Int	Error code.

The output will always consist of three values. The first and third ones are fixed, but the second one will contain the String error message only in case of failure. If the method is successful, the returned value may be of another Data Type.

The Error Code will contain one of the following values:

Value	Code	Meaning
0x0000	SUCCESS	Success response.
0x0100	AUTHENTICATION	User could not be authenticated.
0x0200	AUTHORIZATION	User is not authorized to perform the requested action.
0x0400	NO_EXISTS	The requested resource delhost not exist.
0x0800	ACTION	FIXME
0x1000	XML_RPC_API	FIXME
0x2000	INTERNAL	FIXME

Warning: All methods expect a session string associated to the connected user as the first parameter. It has to be formed with the contents of the ONE_AUTH file, which will be `<username>:<password>` with the default 'core' auth driver.

Warning: Each XML-RPC request has to be authenticated and authorized. See the *Auth Subsystem documentation* for more information.

The information strings returned by the `one.*.info` methods are XML-formatted. The complete XML Schemas (XSD) reference is included at the end of this page. We encourage you to use the `-x` option of the *command line interface* to collect sample outputs from your own infrastructure.

The methods that accept XML templates require the root element to be TEMPLATE. For instance, this template:

```
NAME = abc
MEMORY = 1024
ATT1 = value1
```

Can be also given to OpenNebula with the following XML:

```
<TEMPLATE>
  <NAME>abc</NAME>
  <MEMORY>1024</MEMORY>
  <ATT1>value1</ATT1>
</TEMPLATE>
```

3.1.1 Authorization Requests Reference

For each XML-RPC request, the session token is authenticated, and after that the Request Manager generates an authorization request that can include more than one operation. The following tables document these requests.

onevm

onevm command	XML-RPC Method	Auth. Request
deploy	one.vm.deploy	VM:ADMIN HOST:MANAGE
delete boot shutdown suspend hold stop resume release poweroff reboot	one.vm.action	VM:MANAGE
resched unresched	one.vm.action	VM:ADMIN
migrate	one.vm.migrate	VM:ADMIN HOST:MANAGE
disk-snapshot	one.vm.savedisk	VM:MANAGE IMAGE:CREATE
disk-attach	one.vm.attach	VM:MANAGE IMAGE:USE
disk-detach	one.vm.detach	VM:MANAGE
nic-attach	one.vm.attachnic	VM:MANAGE NET:USE
nic-detach	one.vm.detachnic	VM:MANAGE
create	one.vm.allocate	VM:CREATE IMAGE:USE NET:USE
show	one.vm.info	VM:USE
chown chgrp	one.vm.chown	VM:MANAGE [USER:MANAGE] [GROUP:USE]
chmod	one.vm.chmod	VM:<MANAGEADMIN>
rename	one.vm.rename	VM:MANAGE
snapshot-create	one.vm.snapshotcreate	VM:MANAGE
snapshot-delete	one.vm.snapshotdelete	VM:MANAGE
snapshot-revert	one.vm.snapshotrevert	VM:MANAGE
resize	one.vm.resize	VM:MANAGE
update	one.vm.update	VM:MANAGE
recover	one.vm.recover	VM:ADMIN
list top	one.vmpool.info	VM:USE

Warning: The deploy action requires the user issuing the command to have VM:ADMIN rights. This user will usually be the scheduler with the oneadmin credentials.

The scheduler deploys VMs to the Hosts over which the VM owner has MANAGE rights.

onetemplate

onetemplate command	XML-RPC Method	Auth. Request
update	one.template.update	TEMPLATE:MANAGE
instantiate	one.template.instantiate	TEMPLATE:USE [IMAGE:USE] [NET:USE]
create	one.template.allocate	TEMPLATE:CREATE
clone	one.template.clone	TEMPLATE:CREATE TEMPLATE:USE
delete	one.template.delete	TEMPLATE:MANAGE
show	one.template.info	TEMPLATE:USE
chown chgrp	one.template.chown	TEMPLATE:MANAGE [USER:MANAGE] [GROUP:USE]
chmod	one.template.chmod	TEMPLATE:<MANAGEADMIN
rename	one.template.rename	TEMPLATE:MANAGE
list top	one.templatepool.info	TEMPLATE:USE

onehost

onehost command	XML-RPC Method	Auth. Request
enable disable	one.host.enable	HOST:ADMIN
update	one.host.update	HOST:ADMIN
create	one.host.allocate	HOST:CREATE
delete	one.host.delete	HOST:ADMIN
rename	one.host.rename	HOST:ADMIN
show	one.host.info	HOST:USE
list top	one.hostpool.info	HOST:USE

Warning: onehost sync is not performed by the core, it is done by the ruby command onehost.

onecluster

onecluster command	XML-RPC Method	Auth. Request
create	one.cluster.allocate	CLUSTER:CREATE
delete	one.cluster.delete	CLUSTER:ADMIN
update	one.cluster.update	CLUSTER:MANAGE
addhost	one.cluster.addhost	CLUSTER:ADMIN HOST:ADMIN
delhost	one.cluster.delhost	CLUSTER:ADMIN HOST:ADMIN
adddatastore	one.cluster.adddatastore	CLUSTER:ADMIN DATASTORE:ADMIN
deldatastore	one.cluster.deldatastore	CLUSTER:ADMIN DATASTORE:ADMIN
addvnet	one.cluster.addvnet	CLUSTER:ADMIN NET:ADMIN
delvnet	one.cluster.delvnet	CLUSTER:ADMIN NET:ADMIN
rename	one.cluster.rename	CLUSTER:MANAGE
show	one.cluster.info	CLUSTER:USE
list	one.clusterpool.info	CLUSTER:USE

onegroup

onegroup command	XML-RPC Method	Auth. Request
create	one.group.allocate	GROUP:CREATE
delete	one.group.delete	GROUP:ADMIN
show	one.group.info	GROUP:USE
quota	one.group.quota	GROUP:ADMIN
list	one.grouppool.info	GROUP:USE
•	one.groupquota.info	•
defaultquota	one.groupquota.update group	

onevnet

onevnet command	XML-RPC Method	Auth. Request
addleases	one.vn.addleases	NET:MANAGE
rmleases	one.vn.rmleases	NET:MANAGE
hold	one.vn.hold	NET:MANAGE
release	one.vn.release	NET:MANAGE
update	one.vn.update	NET:MANAGE
create	one.vn.allocate	NET:CREATE
delete	one.vn.delete	NET:MANAGE
show	one.vn.info	NET:USE
chown chgrp	one.vn.chown	NET:MANAGE [USER:MANAGE] [GROUP:USE]
chmod	one.vn.chmod	NET:<MANAGEADMIN>
rename	one.vn.rename	NET:MANAGE
list	one.vnpool.info	NET:USE

oneuser

oneuser command	XML-RPC Method	Auth. Request
create	one.user.allocate	USER:CREATE
delete	one.user.delete	USER:ADMIN
show	one.user.info	USER:USE
passwd	one.user.passwd	USER:MANAGE
update	one.user.update	USER:MANAGE
chauth	one.user.chauth	USER:ADMIN
quota	one.user.quota	USER:ADMIN
chgrp	one.user.chgrp	USER:MANAGE GROUP:USE
addgroup	one.user.addgroup	USER:MANAGE GROUP:MANAGE
delgroup	one.user.delgroup	USER:MANAGE GROUP:MANAGE
encode	•	•
list	one.userpool.info	USER:USE
•	one.userquota.info	•
defaultquota	one.userquota.update group	

onedatastore

oneimage command	XML-RPC Method	Auth. Request
create	one.datastore.allocate	DATASTORE:CREATE
delete	one.datastore.delete	DATASTORE:ADMIN
show	one.datastore.info	DATASTORE:USE
update	one.datastore.update	DATASTORE:MANAGE
rename	one.datastore.rename	DATASTORE:MANAGE
chown chgrp	one.datastore.chown	DATASTORE:MANAGE [USER:MANAGE] [GROUP:USE]
chmod	one.datastore.chmod	DATASTORE:<MANAGE ADMIN>
list	one.datastorepool.info	DATASTORE:USE

oneimage

oneimage command	XML-RPC Method	Auth. Request
persistent nonpersistent	one.image.persistent	IMAGE:MANAGE
enable disable	one.image.enable	IMAGE:MANAGE
chtype	one.image.chtype	IMAGE:MANAGE
update	one.image.update	IMAGE:MANAGE
create	one.image.allocate	IMAGE:CREATE DATASTORE:USE
clone	one.image.clone	IMAGE:CREATE IMAGE:USE
delete	one.image.delete	IMAGE:MANAGE
show	one.image.info	IMAGE:USE
chown chgrp	one.image.chown	IMAGE:MANAGE [USER:MANAGE] [GROUP:USE]
chmod	one.image.chmod	IMAGE:<MANAGE ADMIN>
rename	one.image.rename	IMAGE:MANAGE
list top	one.imagepool.info	IMAGE:USE

oneacl

oneacl command	XML-RPC Method	Auth. Request
create	one.acl.addrule	ACL:MANAGE
delete	one.acl.delrule	ACL:MANAGE
list	one.acl.info	ACL:MANAGE

oneacct

command	XML-RPC Method	Auth. Request
oneacct	one.vmpool.accounting	VM:USE

documents

XML-RPC Method	Auth. Request
one.document.update	DOCUMENT:MANAGE
one.document.allocate	DOCUMENT:CREATE
one.document.delete	DOCUMENT:MANAGE
one.document.info	DOCUMENT:USE
one.document.chown	DOCUMENT:MANAGE [USER:MANAGE] [GROUP:USE]
one.document.chmod	DOCUMENT:<MANAGE ADMIN>
one.document.rename	DOCUMENT:MANAGE
one.documentpool.info	DOCUMENT:USE

system

command	XML-RPC Method	Auth. Request
•	one.system.version	•
•	one.system.config group	

3.1.2 Actions for Templates Management

one.template.allocate

- **Description**: Allocates a new template in OpenNebula.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	String	A string containing the template contents. Syntax can be the usual `attribute=value` or XML.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The allocated resource ID / The error string.
OUT	Int	Error code.

one.template.clone

- **Description**: Clones an existing virtual machine template.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The ID of the template to be cloned.
IN	String	Name for the new template.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The new template ID / The error string.
OUT	Int	Error code.

one.template.delete

- **Description**: Deletes the given template from the pool.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.template.instantiate

- **Description**: Instantiates a new virtual machine from a template.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	String	Name for the new VM instance. If it is an empty string, OpenNebula will assign one automatically.
IN	Boolean	False to create the VM on pending (default), True to create it on hold.
IN	String	A string containing an extra template to be merged with the one being instantiated. It can be empty. Syntax can be the usual `attribute=value` or XML.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The new virtual machine ID / The error string.
OUT	Int	Error code.

one.template.update

- **Description**: Replaces the template contents.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	String	The new template contents. Syntax can be the usual `attribute=value` or XML.
IN	Int	Update type: **0**: replace the whole template. **1**: Merge new template with the existing one.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.template.chmod

- **Description**: Changes the permission bits of a template.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	Int	USER USE bit. If set to -1, it will not change.
IN	Int	USER MANAGE bit. If set to -1, it will not change.
IN	Int	USER ADMIN bit. If set to -1, it will not change.
IN	Int	GROUP USE bit. If set to -1, it will not change.
IN	Int	GROUP MANAGE bit. If set to -1, it will not change.
IN	Int	GROUP ADMIN bit. If set to -1, it will not change.
IN	Int	OTHER USE bit. If set to -1, it will not change.
IN	Int	OTHER MANAGE bit. If set to -1, it will not change.
IN	Int	OTHER ADMIN bit. If set to -1, it will not change.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.template.chown

- **Description**: Changes the ownership of a template.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	Int	The User ID of the new owner. If set to -1, the owner is not changed.
IN	Int	The Group ID of the new group. If set to -1, the group is not changed.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.template.rename

- **Description**: Renames a template.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	String	The new name.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The VM ID / The error string.
OUT	Int	Error code.

one.template.info

- **Description**: Retrieves information for the template.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
OUT	Boolean	true or false whenever is successful or not
OUT	String	The information string / The error string.
OUT	Int	Error code.

one.templatepool.info

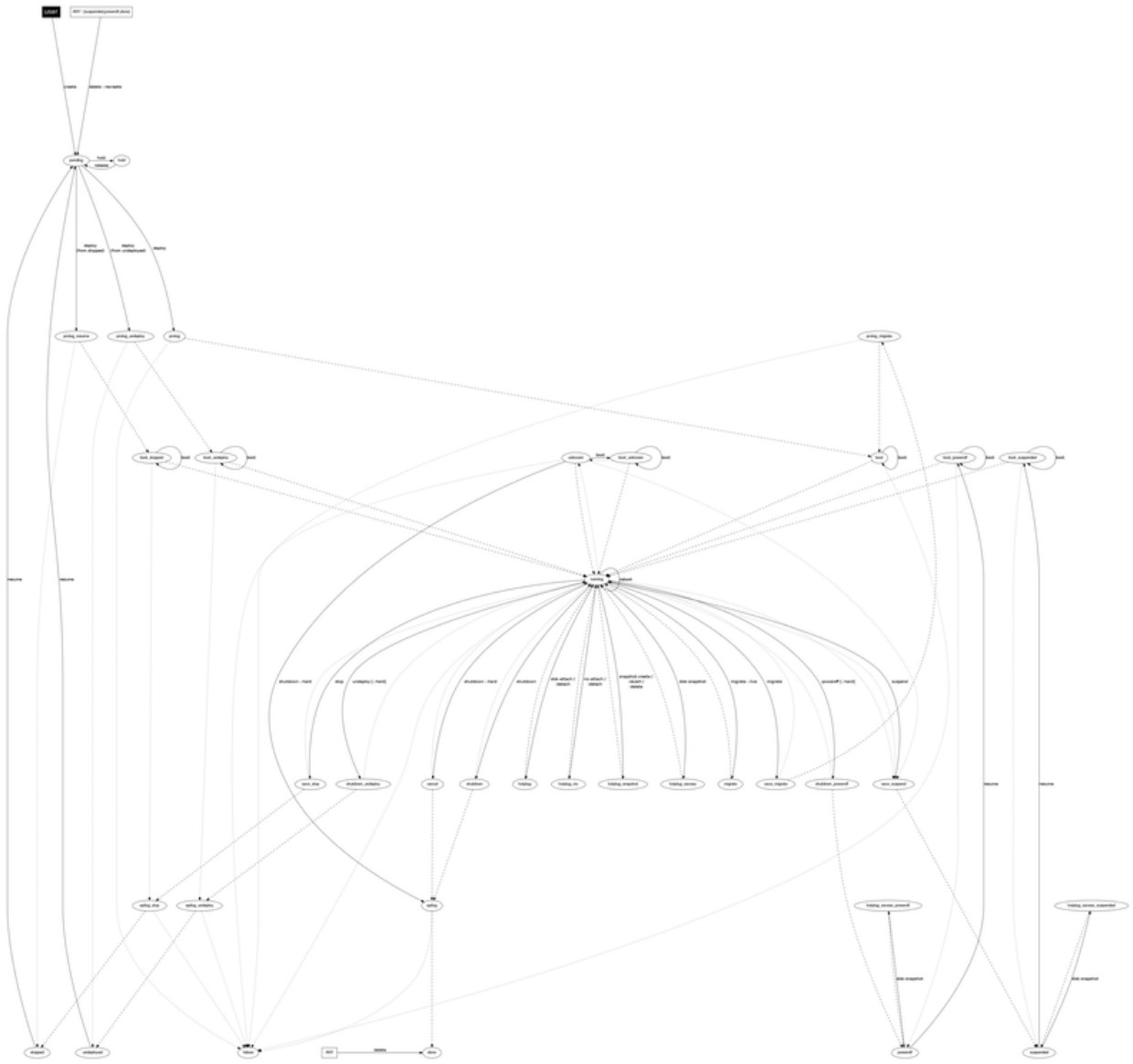
- **Description**: Retrieves information for all or part of the Resources in the pool.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	Filter flag **- < = -3**: Connected user's resources **- -2**: All resources **- -1**: Connected user's and his group's resources **- > = 0**: UID User's Resources
IN	Int	Range start ID. Can be -1.
IN	Int	Range end ID. Can be -1.
OUT	Boolean	true or false whenever is successful or not
OUT	String	The information string / The error string.
OUT	Int	Error code.

The range can be used to retrieve a subset of the pool, from the 'start' to the 'end' ID. To retrieve the complete pool, use `(-1, -1)`; to retrieve all the pool from a specific ID to the last one, use `(<id>, -1)`, and to retrieve the first elements up to an ID, use `(0, <id>)`.

3.1.3 Actions for Virtual Machine Management

The VM Life Cycle is explained in this diagram:.



It contains all the LifeCycleManager states, and the transitions triggered by the onevm commands. It is intended to be consulted by developers.

The simplified diagram used in the *Virtual Machine Instances documentation* uses a smaller number of state names. These names are the ones used by onevm list, e.g. prolog, prolog_migrate and prolog_resume are all presented as `prol`. It is intended as a reference for end-users.

one.vm.allocate

- **Description**: Allocates a new virtual machine in OpenNebula.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	String	A string containing the template for the vm. Syntax can be the usual `attribute=value` or XML.
IN	Boolean	False to create the VM on pending (default), True to create it on hold.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The allocated resource ID / The error string.
OUT	Int	Error code.

one.vm.deploy

- **Description**: initiates the instance of the given vmid on the target host.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	Int	The Host ID of the target host where the VM will be deployed.
IN	Int	The Datastore ID of the target system datastore where the VM will be deployed. It is optional, and can be set to -1 to let OpenNebula choose the datastore.
IN	Boolean	true to enforce the Host capacity is not overcommitted.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The VM ID / The error string.
OUT	Int	Error code.

one.vm.action

- **Description**: submits an action to be performed on a virtual machine.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	String	the action name to be performed, see below.
IN	Int	The object ID.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The VM ID / The error string.
OUT	Int	Error code.

The action String must be one of the following:

- **shutdown**
- **shutdown-hard**
- **hold**
- **release**
- **stop**
- **suspend**
- **resume**
- **boot**

- **delete**
- **delete-recreate**
- **reboot**
- **reboot-hard**
- **resched**
- **unresched**
- **poweroff**
- **poweroff-hard**
- **undeploy**
- **undeploy-hard**

one.vm.migrate

- **Description**: migrates one virtual machine (vid) to the target host (hid).
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	Int	the target host id (hid) where we want to migrate the vm.
IN	Boolean	if true we are indicating that we want livemigration, otherwise false.
IN	Boolean	true to enforce the Host capacity is not overcommitted.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The VM ID / The error string.
OUT	Int	Error code.

one.vm.savedisk

- **Description**: Sets the disk to be saved in the given image.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	Int	Disk ID of the disk we want to save.
IN	String	Name for the new Image where the disk will be saved.
IN	String	Type for the new Image. If it is an empty string, then *the default one* will be used. See the existing types in the *Image template reference*.
IN	Boolean	True to save the disk immediately, false will perform the operation when the VM shuts down.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The new allocated Image ID / The error string.
OUT	Int	Error code.

one.vm.attach

- **Description**: Attaches a new disk to the virtual machine

- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	String	A string containing a single DISK vector attribute. Syntax can be the usual `attribute=value` or XML.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The VM ID / The error string.
OUT	Int	Error code.

one.vm.detach

- **Description**: Detaches a disk from a virtual machine
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	Int	The disk ID.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The VM ID / The error string.
OUT	Int	Error code.

one.vm.attachnic

- **Description**: Attaches a new network interface to the virtual machine
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	String	A string containing a single NIC vector attribute. Syntax can be the usual `attribute=value` or XML.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The VM ID / The error string.
OUT	Int	Error code.

one.vm.detachnic

- **Description**: Detaches a network interface from a virtual machine
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	Int	The nic ID.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The VM ID / The error string.
OUT	Int	Error code.

one.vm.chmod

- **Description**: Changes the permission bits of a virtual machine.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	Int	USER USE bit. If set to -1, it will not change.
IN	Int	USER MANAGE bit. If set to -1, it will not change.
IN	Int	USER ADMIN bit. If set to -1, it will not change.
IN	Int	GROUP USE bit. If set to -1, it will not change.
IN	Int	GROUP MANAGE bit. If set to -1, it will not change.
IN	Int	GROUP ADMIN bit. If set to -1, it will not change.
IN	Int	OTHER USE bit. If set to -1, it will not change.
IN	Int	OTHER MANAGE bit. If set to -1, it will not change.
IN	Int	OTHER ADMIN bit. If set to -1, it will not change.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.vm.chown

- **Description**: Changes the ownership of a virtual machine.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	Int	The User ID of the new owner. If set to -1, the owner is not changed.
IN	Int	The Group ID of the new group. If set to -1, the group is not changed.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.vm.rename

- **Description**: Renames a virtual machine
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	String	The new name.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The VM ID / The error string.
OUT	Int	Error code.

one.vm.snapshotcreate

- **Description**: Creates a new virtual machine snapshot

- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	String	The new snapshot name. It can be empty.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The new snapshot ID / The error string.
OUT	Int	Error code.

one.vm.snapshotrevert

- **Description**: Reverts a virtual machine to a snapshot
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	Int	The snapshot ID.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The VM ID / The error string.
OUT	Int	Error code.

one.vm.snapshotdelete

- **Description**: Deletes a virtual machine snapshot
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	Int	The snapshot ID.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The VM ID / The error string.
OUT	Int	Error code.

one.vm.resize

- **Description**: Changes the capacity of the virtual machine
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	String	Template containing the new capacity elements CPU, VCPU, MEMORY. If one of them is not present, or its value is 0, it will not be resized.
IN	Boolean	true to enforce the Host capacity is not overcommitted. This parameter is only acknoledged for users in the oneadmin group, Host capacity will be always enforced for regular users.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The VM ID / The error string.
OUT	Int	Error code.

one.vm.update

- **Description**: Replaces the **user template** contents.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	String	The new user template contents. Syntax can be the usual `attribute=value` or XML.
IN	Int	Update type: **0**: Replace the whole template. **1**: Merge new template with the existing one.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.vm.recover

- **Description**: Recovers a stuck VM that is waiting for a driver operation. The recovery may be done by failing or succeeding the pending operation. You need to manually check the vm status on the host, to decide if the operation was successful or not.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	Boolean	Recover the VM by succeeding (true) of failing (false) the pending action
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.vm.info

- **Description**: Retrieves information for the virtual machine.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
OUT	Boolean	true or false whenever is successful or not
OUT	String	The information string / The error string.
OUT	Int	Error code.

one.vm.monitoring

- **Description**: Returns the virtual machine monitoring records.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
OUT	Boolean	true or false whenever is successful or not
OUT	String	The monitoring information string / The error string.
OUT	Int	Error code.

The monitoring information returned is a list of VM elements. Each VM element contains the complete xml of the VM with the updated information returned by the poll action.

For example:

```
<MONITORING_DATA>
    <VM>
        ...
        <LAST_POLL>123</LAST_POLL>
        ...
    </VM>
    <VM>
        ...
        <LAST_POLL>456</LAST_POLL>
        ...
    </VM>
</MONITORING_DATA>
```

one.vmpool.info

- **Description**: Retrieves information for all or part of the VMs in the pool.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	Filter flag **- < = -3**: Connected user's resources **- -2**: All resources **- -1**: Connected user's and his group's resources **- > = 0**: UID User's Resources
IN	Int	Range start ID. Can be -1.
IN	Int	Range end ID. Can be -1.
IN	Int	VM state to filter by.
OUT	Boolean	true or false whenever is successful or not
OUT	String	The information string / The error string.
OUT	Int	Error code.

The range can be used to retrieve a subset of the pool, from the 'start' to the 'end' ID. To retrieve the complete pool, use `(-1, -1)`; to retrieve all the pool from a specific ID to the last one, use `(<id>, -1)`, and to retrieve the first elements up to an ID, use `(0, <id>)`.

The state filter can be one of the following:

Value	State
-2	Any state, including DONE
-1	Any state, except DONE
0	INIT
1	PENDING
2	HOLD
3	ACTIVE
4	STOPPED
5	SUSPENDED
6	DONE
7	FAILED

one.vmpool.monitoring

- **Description**: Returns all the virtual machine monitoring records.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	Filter flag **- < = -3**: Connected user's resources **- -2**: All resources **- -1**: Connected user's and his group's resources **- > = 0**: UID User's Resources
OUT	Boolean	true or false whenever is successful or not
OUT	String	The information string / The error string.
OUT	Int	Error code.

See *one.vm.monitoring*.

Sample output:

```
<MONITORING_DATA>
    <VM>
        <ID>0</ID>
        <LAST_POLL>123</LAST_POLL>
        ...
    </VM>
    <VM>
        <ID>0</ID>
        <LAST_POLL>456</LAST_POLL>
        ...
    </VM>
    <VM>
        <ID>3</ID>
        <LAST_POLL>123</LAST_POLL>
        ...
    </VM>
    <VM>
        <ID>3</ID>
        <LAST_POLL>456</LAST_POLL>
        ...
    </VM>
</MONITORING_DATA>
```

one.vmpool.accounting

- **Description**: Returns the virtual machine history records.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	Filter flag **- < = -3**: Connected user's resources **- -2**: All resources **- -1**: Connected user's and his group's resources **- > = 0**: UID User's Resources
IN	Int	Start time for the time interval. Can be -1, in which case the time interval won't have a left boundary.
IN	Int	End time for the time interval. Can be -1, in which case the time interval won't have a right boundary.
OUT	Boolean	true or false whenever is successful or not
OUT	String	The information string / The error string.
OUT	Int	Error code.

The XML output is explained in detail in the *''oneacct" guide*.

3.1.4 Actions for Hosts Management

one.host.allocate

- **Description**: Allocates a new host in OpenNebula
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	String	Hostname of the machine we want to add
IN	String	The name of the information manager (im_mad_name), this values are taken from the oned.conf with the tag name IM_MAD (name)
IN	String	The name of the virtual machine manager mad name (vmm_mad_name), this values are taken from the oned.conf with the tag name VM_MAD (name)
IN	String	The name of the virtual network manager mad name (vnm_mad_name), see the *Networking Subsystem documentation*
IN	Int	The cluster ID. If it is -1, this host won't be added to any cluster.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The allocated Host ID / The error string.
OUT	Int	Error code.

one.host.delete

- **Description**: Deletes the given host from the pool
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.host.enable

- **Description**: Enables or disables the given host

- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The Host ID.
IN	Boolean	Set it to true/false to enable or disable the target Host.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.host.update

- **Description**: Replaces the host's template contents.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	String	The new template contents. Syntax can be the usual `attribute=value` or XML.
IN	Int	Update type: **0**: Replace the whole template. **1**: Merge new template with the existing one.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.host.rename

- **Description**: Renames a host.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	String	The new name.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The VM ID / The error string.
OUT	Int	Error code.

one.host.info

- **Description**: Retrieves information for the host.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
OUT	Boolean	true or false whenever is successful or not
OUT	String	The information string / The error string.
OUT	Int	Error code.

one.host.monitoring

- **Description**: Returns the host monitoring records.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
OUT	Boolean	true or false whenever is successful or not
OUT	String	The monitoring information string / The error string.
OUT	Int	Error code.

The monitoring information returned is a list of HOST elements. Each HOST element contains the complete xml of the host with the updated information returned by the poll action.

For example:

```
<MONITORING_DATA>
    <HOST>
        ...
        <LAST_MON_TIME>123</LAST_MON_TIME>
        ...
    </HOST>
    <HOST>
        ...
        <LAST_MON_TIME>456</LAST_MON_TIME>
        ...
    </HOST>
</MONITORING_DATA>
```

one.hostpool.info

- **Description**: Retrieves information for all the hosts in the pool.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
OUT	Boolean	true or false whenever is successful or not
OUT	String	The information string / The error string.
OUT	Int	Error code.

one.hostpool.monitoring

- **Description**: Returns all the host monitoring records.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
OUT	Boolean	true or false whenever is successful or not
OUT	String	The information string / The error string.
OUT	Int	Error code.

Sample output:

```
<MONITORING_DATA>
    <HOST>
        <ID>0</ID>
        <LAST_MON_TIME>123</LAST_MON_TIME>
        ...
    </HOST>
    <HOST>
        <ID>0</ID>
        <LAST_MON_TIME>456</LAST_MON_TIME>
        ...
    </HOST>
    <HOST>
        <ID>3</ID>
        <LAST_MON_TIME>123</LAST_MON_TIME>
        ...
    </HOST>
    <HOST>
        <ID>3</ID>
        <LAST_MON_TIME>456</LAST_MON_TIME>
        ...
    </HOST>
</MONITORING_DATA>
```

3.1.5 Actions for Cluster Management

one.cluster.allocate

- **Description**: Allocates a new cluster in OpenNebula.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	String	Name for the new cluster.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The allocated cluster ID / The error string.
OUT	Int	Error code.

one.cluster.delete

- **Description**: Deletes the given cluster from the pool.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.cluster.update

- **Description**: Replaces the cluster template contents.

- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	String	The new template contents. Syntax can be the usual `attribute=value` or XML.
IN	Int	Update type: **0**: Replace the whole template. **1**: Merge new template with the existing one.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.cluster.addhost

- **Description**: Adds a host to the given cluster.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The cluster ID.
IN	Int	The host ID.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.cluster.delhost

- **Description**: Removes a host from the given cluster.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The cluster ID.
IN	Int	The host ID.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.cluster.adddatastore

- **Description**: Adds a datastore to the given cluster.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The cluster ID.
IN	Int	The datastore ID.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.cluster.deldatastore

- **Description**: Removes a datastore from the given cluster.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The cluster ID.
IN	Int	The datastore ID.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.cluster.addvnet

- **Description**: Adds a vnet to the given cluster.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The cluster ID.
IN	Int	The vnet ID.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.cluster.delvnet

- **Description**: Removes a vnet from the given cluster.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The cluster ID.
IN	Int	The vnet ID.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.cluster.rename

- **Description**: Renames a cluster.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	String	The new name.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The VM ID / The error string.
OUT	Int	Error code.

one.cluster.info

- **Description**: Retrieves information for the cluster.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
OUT	Boolean	true or false whenever is successful or not
OUT	String	The information string / The error string.
OUT	Int	Error code.

one.clusterpool.info

- **Description**: Retrieves information for all the clusters in the pool.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
OUT	Boolean	true or false whenever is successful or not
OUT	String	The information string / The error string.
OUT	Int	Error code.

3.1.6 Actions for Virtual Network Management

one.vn.allocate

- **Description**: Allocates a new virtual network in OpenNebula.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	String	A string containing the template of the virtual network. Syntax can be the usual `attribute=value` or XML.
IN	Int	The cluster ID. If it is -1, this virtual network won't be added to any cluster.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The allocated resource ID / The error string.
OUT	Int	Error code.

one.vn.delete

- **Description**: Deletes the given virtual network from the pool.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.vn.addleases

- **Description**: Adds a new lease to the virtual network. Only available for FIXED networks.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	String	template of the lease to add. Syntax can be the usual `attribute=value` or XML, see below.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

Examples of valid templates:

```
LEASES=[IP=192.168.0.5]

LEASES=[IP=192.168.0.5, MAC=50:20:20:20:20:20]

<TEMPLATE>
  <LEASES>
    <IP>192.168.0.5</IP>
  </LEASES>
</TEMPLATE>

<TEMPLATE>
  <LEASES>
    <IP>192.168.0.5</IP>
    <MAC>MAC=50:20:20:20:20:20</MAC>
  </LEASES>
</TEMPLATE>
```

one.vn.rmleases

- **Description**: Removes a lease from the virtual network. Only available for FIXED networks.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	String	template of the lease to remove. Syntax can be the usual `attribute=value` or XML.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.vn.hold

- **Description**: Holds a virtual network Lease as used.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	String	template of the lease to hold, e.g. `LEASES=[IP=192.168.0.5]`.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.vn.release

- **Description**: Releases a virtual network Lease on hold.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	String	template of the lease to release, e.g. `LEASES=[IP=192.168.0.5]`.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.vn.update

- **Description**: Replaces the virtual network template contents.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	String	The new template contents. Syntax can be the usual `attribute=value` or XML.
IN	Int	Update type: **0**: Replace the whole template. **1**: Merge new template with the existing one.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.vn.chmod

- **Description**: Changes the permission bits of a virtual network.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	Int	USER USE bit. If set to -1, it will not change.
IN	Int	USER MANAGE bit. If set to -1, it will not change.
IN	Int	USER ADMIN bit. If set to -1, it will not change.
IN	Int	GROUP USE bit. If set to -1, it will not change.
IN	Int	GROUP MANAGE bit. If set to -1, it will not change.
IN	Int	GROUP ADMIN bit. If set to -1, it will not change.
IN	Int	OTHER USE bit. If set to -1, it will not change.
IN	Int	OTHER MANAGE bit. If set to -1, it will not change.
IN	Int	OTHER ADMIN bit. If set to -1, it will not change.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.vn.chown

- **Description**: Changes the ownership of a virtual network.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	Int	The User ID of the new owner. If set to -1, the owner is not changed.
IN	Int	The Group ID of the new group. If set to -1, the group is not changed.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.vn.rename

- **Description**: Renames a virtual network.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	String	The new name.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The VM ID / The error string.
OUT	Int	Error code.

one.vn.info

- **Description**: Retrieves information for the virtual network.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
OUT	Boolean	true or false whenever is successful or not
OUT	String	The information string / The error string.
OUT	Int	Error code.

one.vnpool.info

- **Description**: Retrieves information for all or part of the virtual networks in the pool.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	Filter flag **- < = -3**: Connected user's resources **- -2**: All resources **- -1**: Connected user's and his group's resources **- > = 0**: UID User's Resources
IN	Int	Range start ID. Can be -1.
IN	Int	Range end ID. Can be -1.
OUT	Boolean	true or false whenever is successful or not
OUT	String	The information string / The error string.
OUT	Int	Error code.

The range can be used to retrieve a subset of the pool, from the 'start' to the 'end' ID. To retrieve the complete pool, use `(-1, -1)`; to retrieve all the pool from a specific ID to the last one, use `(<id>, -1)`, and to retrieve the first elements up to an ID, use `(0, <id>)`.

3.1.7 Actions for Datastore Management

one.datastore.allocate

- **Description**: Allocates a new datastore in OpenNebula.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	String	A string containing the template of the datastore. Syntax can be the usual `attribute=value` or XML.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The allocated resource ID / The error string.
OUT	Int	Error code.

one.datastore.delete

- **Description**: Deletes the given datastore from the pool.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.datastore.update

- **Description**: Replaces the datastore template contents.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	String	The new template contents. Syntax can be the usual `attribute=value` or XML.
IN	Int	Update type: **0**: Replace the whole template. **1**: Merge new template with the existing one.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.datastore.chmod

- **Description**: Changes the permission bits of a datastore.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	Int	USER USE bit. If set to -1, it will not change.
IN	Int	USER MANAGE bit. If set to -1, it will not change.
IN	Int	USER ADMIN bit. If set to -1, it will not change.
IN	Int	GROUP USE bit. If set to -1, it will not change.
IN	Int	GROUP MANAGE bit. If set to -1, it will not change.
IN	Int	GROUP ADMIN bit. If set to -1, it will not change.
IN	Int	OTHER USE bit. If set to -1, it will not change.
IN	Int	OTHER MANAGE bit. If set to -1, it will not change.
IN	Int	OTHER ADMIN bit. If set to -1, it will not change.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.datastore.chown

- **Description**: Changes the ownership of a datastore.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	Int	The User ID of the new owner. If set to -1, the owner is not changed.
IN	Int	The Group ID of the new group. If set to -1, the group is not changed.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.datastore.rename

- **Description**: Renames a datastore.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	String	The new name.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The VM ID / The error string.
OUT	Int	Error code.

one.datastore.info

- **Description**: Retrieves information for the datastore.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
OUT	Boolean	true or false whenever is successful or not
OUT	String	The information string / The error string.
OUT	Int	Error code.

one.datastorepool.info

- **Description**: Retrieves information for all or part of the datastores in the pool.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
OUT	Boolean	true or false whenever is successful or not
OUT	String	The information string / The error string.
OUT	Int	Error code.

3.1.8 Actions for Image Management

one.image.allocate

- **Description**: Allocates a new image in OpenNebula.

- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	String	A string containing the template of the image. Syntax can be the usual `attribute=value` or XML.
IN	Int	The datastore ID.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The allocated resource ID / The error string.
OUT	Int	Error code.

one.image.clone

- **Description**: Clones an existing image.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The ID of the image to be cloned.
IN	String	Name for the new image.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The new image ID / The error string.
OUT	Int	Error code.

one.image.delete

- **Description**: Deletes the given image from the pool.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.image.enable

- **Description**: Enables or disables an image.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The Image ID.
IN	Boolean	True for enabling, false for disabling.
OUT	Boolean	true or false whenever is successful or not.
OUT	Int/String	The Image ID / The error string.
OUT	Int	Error code.

one.image.persistent

- **Description**: Sets the Image as persistent or not persistent.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The Image ID.
IN	Boolean	True for persistent, false for non-persisent.
OUT	Boolean	true or false whenever is successful or not.
OUT	Int/String	The Image ID / The error string.
OUT	Int	Error code.

one.image.chtype

- **Description**: Changes the type of an Image.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The Image ID.
IN	String	New type for the Image. See the existing types in the *Image template reference*.
OUT	Boolean	true or false whenever is successful or not.
OUT	Int/String	The Image ID / The error string.
OUT	Int	Error code.

one.image.update

- **Description**: Replaces the image template contents.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	String	The new template contents. Syntax can be the usual `attribute=value` or XML.
IN	Int	Update type: **0**: Replace the whole template. **1**: Merge new template with the existing one.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.image.chmod

- **Description**: Changes the permission bits of an image.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	Int	USER USE bit. If set to -1, it will not change.
IN	Int	USER MANAGE bit. If set to -1, it will not change.
IN	Int	USER ADMIN bit. If set to -1, it will not change.
IN	Int	GROUP USE bit. If set to -1, it will not change.
IN	Int	GROUP MANAGE bit. If set to -1, it will not change.
IN	Int	GROUP ADMIN bit. If set to -1, it will not change.
IN	Int	OTHER USE bit. If set to -1, it will not change.
IN	Int	OTHER MANAGE bit. If set to -1, it will not change.
IN	Int	OTHER ADMIN bit. If set to -1, it will not change.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.image.chown

- **Description**: Changes the ownership of an image.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	Int	The User ID of the new owner. If set to -1, the owner is not changed.
IN	Int	The Group ID of the new group. If set to -1, the group is not changed.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.image.rename

- **Description**: Renames an image.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	String	The new name.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The VM ID / The error string.
OUT	Int	Error code.

one.image.info

- **Description**: Retrieves information for the image.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
OUT	Boolean	true or false whenever is successful or not
OUT	String	The information string / The error string.
OUT	Int	Error code.

one.imagepool.info

- **Description**: Retrieves information for all or part of the images in the pool.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	Filter flag **- < = -3**: Connected user's resources **- -2**: All resources **- -1**: Connected user's and his group's resources **- > = 0**: UID User's Resources
IN	Int	Range start ID. Can be -1.
IN	Int	Range end ID. Can be -1.
OUT	Boolean	true or false whenever is successful or not
OUT	String	The information string / The error string.
OUT	Int	Error code.

The range can be used to retrieve a subset of the pool, from the 'start' to the 'end' ID. To retrieve the complete pool, use `(-1, -1)`; to retrieve all the pool from a specific ID to the last one, use `(<id>, -1)`, and to retrieve the first elements up to an ID, use `(0, <id>)`.

3.1.9 Actions for User Management

one.user.allocate

- **Description**: Allocates a new user in OpenNebula
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	String	username for the new user
IN	String	password for the new user
IN	String	authentication driver for the new user. If it is an empty string, then the default ('core') is used
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The allocated User ID / The error string.
OUT	Int	Error code.

one.user.delete

- **Description**: Deletes the given user from the pool.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.user.passwd

- **Description**: Changes the password for the given user.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	String	The new password
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The User ID / The error string.
OUT	Int	Error code.

one.user.update

- **Description**: Replaces the user template contents.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	String	The new template contents. Syntax can be the usual `attribute=value` or XML.
IN	Int	Update type: **0**: Replace the whole template. **1**: Merge new template with the existing one.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.user.chauth

- **Description**: Changes the authentication driver and the password for the given user.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	String	The new authentication driver.
IN	String	The new password. If it is an empty string, the password is not changed.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The User ID / The error string.
OUT	Int	Error code.

one.user.quota

- **Description**: Sets the user quota limits.

- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	String	The new quota template contents. Syntax can be the usual `attribute=value` or XML.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.user.chgrp

- **Description**: Changes the group of the given user.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The User ID.
IN	Int	The Group ID of the new group.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The User ID / The error string.
OUT	Int	Error code.

one.user.addgroup

- **Description**: Adds the User to a secondary group.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The User ID.
IN	Int	The Group ID of the new group.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The User ID / The error string.
OUT	Int	Error code.

one.user.delgroup

- **Description**: Removes the User from a secondary group
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The User ID.
IN	Int	The Group ID.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The User ID / The error string.
OUT	Int	Error code.

one.user.info

- **Description**: Retrieves information for the user.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID. If it is -1, then the connected user's own info info is returned
OUT	Boolean	true or false whenever is successful or not
OUT	String	The information string / The error string.
OUT	Int	Error code.

one.userpool.info

- **Description**: Retrieves information for all the users in the pool.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
OUT	Boolean	true or false whenever is successful or not
OUT	String	The information string / The error string.
OUT	Int	Error code.

one.userquota.info

- **Description**: Returns the default user quota limits.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
OUT	Boolean	true or false whenever is successful or not
OUT	String	The quota template contents / The error string.
OUT	Int	Error code.

one.userquota.update

- **Description**: Updates the default user quota limits.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	String	The new quota template contents. Syntax can be the usual `attribute=value` or XML.
OUT	Boolean	true or false whenever is successful or not
OUT	String	The quota template contents / The error string.
OUT	Int	Error code.

3.1.10 Actions for Group Management

one.group.allocate

- **Description**: Allocates a new group in OpenNebula.

- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	String	Name for the new group.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The allocated Group ID / The error string.
OUT	Int	Error code.

one.group.delete

- **Description**: Deletes the given group from the pool.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.group.info

- **Description**: Retrieves information for the group.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID. If it is -1, then the connected user's group info info is returned
OUT	Boolean	true or false whenever is successful or not
OUT	String	The information string / The error string.
OUT	Int	Error code.

one.group.quota

- **Description**: Sets the group quota limits.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	String	The new quota template contents. Syntax can be the usual `attribute=value` or XML.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.grouppool.info

- **Description**: Retrieves information for all the groups in the pool.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
OUT	Boolean	true or false whenever is successful or not
OUT	String	The information string / The error string.
OUT	Int	Error code.

one.groupquota.info

- **Description**: Returns the default group quota limits.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
OUT	Boolean	true or false whenever is successful or not
OUT	String	The quota template contents / The error string.
OUT	Int	Error code.

one.groupquota.update

- **Description**: Updates the default group quota limits.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	String	The new quota template contents. Syntax can be the usual `attribute=value` or XML.
OUT	Boolean	true or false whenever is successful or not
OUT	String	The quota template contents / The error string.
OUT	Int	Error code.

3.1.11 Actions for ACL Rules Management

one.acl.addrule

- **Description**: Adds a new ACL rule.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	String	User component of the new rule. A string containing a hex number.
IN	String	Resource component of the new rule. A string containing a hex number.
IN	String	Rights component of the new rule. A string containing a hex number.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The allocated ACL rule ID / The error string.
OUT	Int	Error code.

To build the hex. numbers required to create a new rule we recommend you to read the ruby or java code.

one.acl.delrule

- **Description**: Deletes an ACL rule.

- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	ACL rule ID.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The ACL rule ID / The error string.
OUT	Int	Error code.

one.acl.info

- **Description**: Returns the complete ACL rule set.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	ACL rule ID.
OUT	Boolean	true or false whenever is successful or not
OUT	String	The information string / The error string.
OUT	Int	Error code.

3.1.12 Actions for Document Management

one.document.allocate

- **Description**: Allocates a new document in OpenNebula.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	String	A string containing the document template contents. Syntax can be the usual `attribute=value` or XML.
IN	Int	The document type (*).
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The allocated resource ID / The error string.
OUT	Int	Error code.

(*) Type is an integer value used to allow dynamic pools compartmentalization.

Let's say you want to store documents representing Chef recipes, and EC2 security groups; you would allocate documents of each kind with a different type. This type is then used in the one.documentpool.info method to filter the results.

one.document.clone

- **Description**: Clones an existing virtual machine document.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The ID of the document to be cloned.
IN	String	Name for the new document.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The new document ID / The error string.
OUT	Int	Error code.

one.document.delete

- **Description**: Deletes the given document from the pool.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.document.update

- **Description**: Replaces the document template contents.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	String	The new document template contents. Syntax can be the usual `attribute=value` or XML.
IN	Int	Update type: **0**: Replace the whole template. **1**: Merge new template with the existing one.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.document.chmod

- **Description**: Changes the permission bits of a document.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	Int	USER USE bit. If set to -1, it will not change.
IN	Int	USER MANAGE bit. If set to -1, it will not change.
IN	Int	USER ADMIN bit. If set to -1, it will not change.
IN	Int	GROUP USE bit. If set to -1, it will not change.
IN	Int	GROUP MANAGE bit. If set to -1, it will not change.
IN	Int	GROUP ADMIN bit. If set to -1, it will not change.
IN	Int	OTHER USE bit. If set to -1, it will not change.
IN	Int	OTHER MANAGE bit. If set to -1, it will not change.
IN	Int	OTHER ADMIN bit. If set to -1, it will not change.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.document.chown

- **Description**: Changes the ownership of a document.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	Int	The User ID of the new owner. If set to -1, the owner is not changed.
IN	Int	The Group ID of the new group. If set to -1, the group is not changed.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The resource ID / The error string.
OUT	Int	Error code.

one.document.rename

- **Description**: Renames a document.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
IN	String	The new name.
OUT	Boolean	true or false whenever is successful or not
OUT	Int/String	The VM ID / The error string.
OUT	Int	Error code.

one.document.info

- **Description**: Retrieves information for the document.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	The object ID.
OUT	Boolean	true or false whenever is successful or not
OUT	String	The information string / The error string.
OUT	Int	Error code.

one.documentpool.info

- **Description**: Retrieves information for all or part of the Resources in the pool.
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
IN	Int	Filter flag **- < = -3**: Connected user's resources **- -2**: All resources **- -1**: Connected user's and his group's resources **- > = 0**: UID User's Resources
IN	Int	Range start ID. Can be -1.
IN	Int	Range end ID. Can be -1.
IN	Int	The document type.
OUT	Boolean	true or false whenever is successful or not
OUT	String	The information string / The error string.
OUT	Int	Error code.

The range can be used to retrieve a subset of the pool, from the 'start' to the 'end' ID. To retrieve the complete pool, use `(-1, -1)`; to retrieve all the pool from a specific ID to the last one, use `(<id>, -1)`, and to retrieve the first elements up to an ID, use `(0, <id>)`.

3.1.13 System Methods

one.system.version

- **Description**: Returns the OpenNebula core version
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
OUT	Boolean	true or false whenever is successful or not
OUT	String	The OpenNebula version, e.g. `4.4.0`
OUT	Int	Error code.

one.system.config

- **Description**: Returns the OpenNebula configuration
- **Parameters**

Type	Data Type	Description
IN	String	The session string.
OUT	Boolean	true or false whenever is successful or not
OUT	String	The loaded oned.conf file, in XML form
OUT	Int	Error code.

3.1.14 XSD Reference

The XML schemas describe the XML returned by the one.*.info methods

Schemas for Cluster

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
  targetNamespace="http://opennebula.org/XMLSchema" xmlns="http://opennebula.org/XMLSchema">
  <xs:element name="CLUSTER">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ID" type="xs:integer"/>
        <xs:element name="NAME" type="xs:string"/>
        <xs:element name="HOSTS">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="ID" type="xs:integer" minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
              </xs:complexType>
        </xs:element>
        <xs:element name="DATASTORES">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="ID" type="xs:integer" minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
              </xs:complexType>
        </xs:element>
        <xs:element name="VNETS">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="ID" type="xs:integer" minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
              </xs:complexType>
        </xs:element>
        <xs:element name="TEMPLATE" type="xs:anyType"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
  targetNamespace="http://opennebula.org/XMLSchema" xmlns="http://opennebula.org/XMLSchema">
  <xs:include schemaLocation="cluster.xsd"/>
  <xs:element name="CLUSTER_POOL">
    <xs:complexType>
      <xs:sequence maxOccurs="1" minOccurs="1">
        <xs:element ref="CLUSTER" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Schemas for Datastore

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://opennebula.org/XMLSchema" elemer
  <xs:element name="DATASTORE">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ID" type="xs:integer"/>
        <xs:element name="UID" type="xs:integer"/>
        <xs:element name="GID" type="xs:integer"/>
        <xs:element name="UNAME" type="xs:string"/>
        <xs:element name="GNAME" type="xs:string"/>
        <xs:element name="NAME" type="xs:string"/>
        <xs:element name="PERMISSIONS" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="OWNER_U" type="xs:integer"/>
              <xs:element name="OWNER_M" type="xs:integer"/>
              <xs:element name="OWNER_A" type="xs:integer"/>
              <xs:element name="GROUP_U" type="xs:integer"/>
              <xs:element name="GROUP_M" type="xs:integer"/>
              <xs:element name="GROUP_A" type="xs:integer"/>
              <xs:element name="OTHER_U" type="xs:integer"/>
              <xs:element name="OTHER_M" type="xs:integer"/>
              <xs:element name="OTHER_A" type="xs:integer"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>

        <xs:element name="DS_MAD" type="xs:string"/>
        <xs:element name="TM_MAD" type="xs:string"/>
        <xs:element name="BASE_PATH" type="xs:string"/>
        <xs:element name="TYPE" type="xs:integer"/>
        <xs:element name="DISK_TYPE" type="xs:integer"/>
        <xs:element name="CLUSTER_ID" type="xs:integer"/>
        <xs:element name="CLUSTER" type="xs:string"/>
        <xs:element name="TOTAL_MB" type="xs:integer"/>
        <xs:element name="FREE_MB" type="xs:integer"/>
        <xs:element name="USED_MB" type="xs:integer"/>
        <xs:element name="IMAGES">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="ID" type="xs:integer" minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
              </xs:complexType>
        </xs:element>
        <xs:element name="TEMPLATE" type="xs:anyType"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
  targetNamespace="http://opennebula.org/XMLSchema" xmlns="http://opennebula.org/XMLSchema">
  <xs:include schemaLocation="datastore.xsd"/>
  <xs:element name="DATASTORE_POOL">
    <xs:complexType>
```

```
      <xs:sequence maxOccurs="1" minOccurs="1">
        <xs:element ref="DATASTORE" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Schemas for Group

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
  targetNamespace="http://opennebula.org/XMLSchema" xmlns="http://opennebula.org/XMLSchema">
  <xs:element name="GROUP">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ID" type="xs:integer"/>
        <xs:element name="NAME" type="xs:string"/>
        <xs:element name="USERS">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="ID" type="xs:integer" minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
              </xs:complexType>
        </xs:element>
        <xs:element name="DATASTORE_QUOTA" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="DATASTORE" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="ID" type="xs:string"/>
                  <xs:element name="IMAGES" type="xs:string"/>
                  <xs:element name="IMAGES_USED" type="xs:string"/>
                  <xs:element name="SIZE" type="xs:string"/>
                  <xs:element name="SIZE_USED" type="xs:string"/>
                </xs:sequence>
              </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="NETWORK_QUOTA" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="NETWORK" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="ID" type="xs:string"/>
                  <xs:element name="LEASES" type="xs:string"/>
                  <xs:element name="LEASES_USED" type="xs:string"/>
                </xs:sequence>
              </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="VM_QUOTA" minOccurs="0" maxOccurs="1">
```

```
  <xs:complexType>
    <xs:sequence>
      <xs:element name="VM" minOccurs="0" maxOccurs="1">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="CPU" type="xs:string"/>
            <xs:element name="CPU_USED" type="xs:string"/>
            <xs:element name="MEMORY" type="xs:string"/>
            <xs:element name="MEMORY_USED" type="xs:string"/>
            <xs:element name="VMS" type="xs:string"/>
            <xs:element name="VMS_USED" type="xs:string"/>
            <xs:element name="VOLATILE_SIZE" type="xs:string"/>
            <xs:element name="VOLATILE_SIZE_USED" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="IMAGE_QUOTA" minOccurs="0" maxOccurs="1">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="IMAGE" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="ID" type="xs:string"/>
            <xs:element name="RVMS" type="xs:string"/>
            <xs:element name="RVMS_USED" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="DEFAULT_GROUP_QUOTAS">
  <xs:complexType>
    <xs:sequence>
        <xs:element name="DATASTORE_QUOTA" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="DATASTORE" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="ID" type="xs:string"/>
                  <xs:element name="IMAGES" type="xs:string"/>
                  <xs:element name="IMAGES_USED" type="xs:string"/>
                  <xs:element name="SIZE" type="xs:string"/>
                  <xs:element name="SIZE_USED" type="xs:string"/>
                </xs:sequence>
              </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="NETWORK_QUOTA" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="NETWORK" minOccurs="0" maxOccurs="unbounded">
```

```
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element name="ID" type="xs:string"/>
                        <xs:element name="LEASES" type="xs:string"/>
                        <xs:element name="LEASES_USED" type="xs:string"/>
                      </xs:sequence>
                    </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="VM_QUOTA" minOccurs="0" maxOccurs="1">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="VM" minOccurs="0" maxOccurs="1">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="CPU" type="xs:string"/>
                          <xs:element name="CPU_USED" type="xs:string"/>
                          <xs:element name="MEMORY" type="xs:string"/>
                          <xs:element name="MEMORY_USED" type="xs:string"/>
                          <xs:element name="VMS" type="xs:string"/>
                          <xs:element name="VMS_USED" type="xs:string"/>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="IMAGE_QUOTA" minOccurs="0" maxOccurs="1">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="IMAGE" minOccurs="0" maxOccurs="unbounded">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="ID" type="xs:string"/>
                          <xs:element name="RVMS" type="xs:string"/>
                          <xs:element name="RVMS_USED" type="xs:string"/>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
  targetNamespace="http://opennebula.org/XMLSchema" xmlns="http://opennebula.org/XMLSchema">
  <xs:element name="GROUP_POOL">
    <xs:complexType>
      <xs:sequence maxOccurs="1" minOccurs="1">
```

```
<xs:element name="GROUP" maxOccurs="unbounded" minOccurs="0">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="ID" type="xs:integer"/>
      <xs:element name="NAME" type="xs:string"/>
      <xs:element name="USERS">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="ID" type="xs:integer" minOccurs="0" maxOccurs="unbounded"
              </xs:sequence>
            </xs:complexType>
      </xs:element>
      <xs:element name="DATASTORE_QUOTA" minOccurs="0" maxOccurs="1">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="DATASTORE" minOccurs="0" maxOccurs="unbounded">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="ID" type="xs:string"/>
                <xs:element name="IMAGES" type="xs:string"/>
                <xs:element name="IMAGES_USED" type="xs:string"/>
                <xs:element name="SIZE" type="xs:string"/>
                <xs:element name="SIZE_USED" type="xs:string"/>
              </xs:sequence>
            </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="NETWORK_QUOTA" minOccurs="0" maxOccurs="1">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="NETWORK" minOccurs="0" maxOccurs="unbounded">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="ID" type="xs:string"/>
                <xs:element name="LEASES" type="xs:string"/>
                <xs:element name="LEASES_USED" type="xs:string"/>
              </xs:sequence>
            </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="VM_QUOTA" minOccurs="0" maxOccurs="1">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="VM" minOccurs="0" maxOccurs="1">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="CPU" type="xs:string"/>
                  <xs:element name="CPU_USED" type="xs:string"/>
                  <xs:element name="MEMORY" type="xs:string"/>
                  <xs:element name="MEMORY_USED" type="xs:string"/>
                  <xs:element name="VMS" type="xs:string"/>
                  <xs:element name="VMS_USED" type="xs:string"/>
                  <xs:element name="VOLATILE_SIZE" type="xs:string"/>
                  <xs:element name="VOLATILE_SIZE_USED" type="xs:string"/>
```

```
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="IMAGE_QUOTA" minOccurs="0" maxOccurs="1">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="IMAGE" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="ID" type="xs:string"/>
                  <xs:element name="RVMS" type="xs:string"/>
                  <xs:element name="RVMS_USED" type="xs:string"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="DEFAULT_GROUP_QUOTAS" minOccurs="1" maxOccurs="1">
  <xs:complexType>
    <xs:sequence>
        <xs:element name="DATASTORE_QUOTA" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="DATASTORE" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="ID" type="xs:string"/>
                  <xs:element name="IMAGES" type="xs:string"/>
                  <xs:element name="IMAGES_USED" type="xs:string"/>
                  <xs:element name="SIZE" type="xs:string"/>
                  <xs:element name="SIZE_USED" type="xs:string"/>
                </xs:sequence>
              </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="NETWORK_QUOTA" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="NETWORK" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="ID" type="xs:string"/>
                  <xs:element name="LEASES" type="xs:string"/>
                  <xs:element name="LEASES_USED" type="xs:string"/>
                </xs:sequence>
              </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
```

```
                    </xs:element>
                    <xs:element name="VM_QUOTA" minOccurs="0" maxOccurs="1">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="VM" minOccurs="0" maxOccurs="1">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="CPU" type="xs:string"/>
                                <xs:element name="CPU_USED" type="xs:string"/>
                                <xs:element name="MEMORY" type="xs:string"/>
                                <xs:element name="MEMORY_USED" type="xs:string"/>
                                <xs:element name="VMS" type="xs:string"/>
                                <xs:element name="VMS_USED" type="xs:string"/>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                    <xs:element name="IMAGE_QUOTA" minOccurs="0" maxOccurs="1">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="IMAGE" minOccurs="0" maxOccurs="unbounded">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="ID" type="xs:string"/>
                                <xs:element name="RVMS" type="xs:string"/>
                                <xs:element name="RVMS_USED" type="xs:string"/>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:schema>
```

Schemas for Host

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://opennebula.org/XMLSchema" elemen
  <xs:element name="HOST">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ID" type="xs:integer"/>
        <xs:element name="NAME" type="xs:string"/>
        <!-- STATE values

          INIT                = 0  Initial state for enabled hosts
          MONITORING_MONITORED = 1  Monitoring the host (from monitored)
          MONITORED           = 2  The host has been successfully monitored
          ERROR               = 3  An error ocurrer while monitoring the host
```

```
  DISABLED             = 4  The host is disabled won't be monitored
  MONITORING_ERROR     = 5  Monitoring the host (from error)
  MONITORING_INIT      = 6  Monitoring the host (from init)
  MONITORING_DISABLED  = 7  Monitoring the host (from disabled)
-->
<xs:element name="STATE" type="xs:integer"/>
<xs:element name="IM_MAD" type="xs:string"/>
<xs:element name="VM_MAD" type="xs:string"/>
<xs:element name="VN_MAD" type="xs:string"/>
<xs:element name="LAST_MON_TIME" type="xs:integer"/>
<xs:element name="CLUSTER_ID" type="xs:integer"/>
<xs:element name="CLUSTER" type="xs:string"/>
<xs:element name="HOST_SHARE">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="DISK_USAGE" type="xs:integer"/>
      <xs:element name="MEM_USAGE" type="xs:integer"/>
      <!-- ^^ KB, Usage of MEMORY calculated by ONE as the summatory MEMORY requested by all
      <xs:element name="CPU_USAGE" type="xs:integer"/>
      <!-- ^^ Percentage, Usage of CPU calculated by ONE as the summatory CPU requested by al
      <xs:element name="MAX_DISK" type="xs:integer"/>
      <xs:element name="MAX_MEM" type="xs:integer"/>
      <!-- ^^ KB, Total memory in the host   -->
      <xs:element name="MAX_CPU" type="xs:integer"/>
      <!-- ^^ Percentage, Total CPU in the host (# cores * 100)  -->
      <xs:element name="FREE_DISK" type="xs:integer"/>
      <xs:element name="FREE_MEM" type="xs:integer"/>
      <!-- ^^ KB, Free MEMORY returned by the probes   -->
      <xs:element name="FREE_CPU" type="xs:integer"/>
      <!-- ^^ Percentage, Free CPU as returned by the probes   -->
      <xs:element name="USED_DISK" type="xs:integer"/>
      <xs:element name="USED_MEM" type="xs:integer"/>
      <!-- ^^ KB, Memory used by all host processes (including VMs) over a total of MAX_MEM
      <xs:element name="USED_CPU" type="xs:integer"/>
      <!-- ^^ Percentage of CPU used by all host processes (including VMs) over a total of #
      <xs:element name="RUNNING_VMS" type="xs:integer"/>
      <xs:element name="DATASTORES">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="DS" minOccurs="0" maxOccurs="unbounded">
                      <xs:complexType>
                        <xs:all>
                          <xs:element name="ID" type="xs:integer"/>
                          <xs:element name="FREE_MB" type="xs:integer"/>
                          <xs:element name="TOTAL_MB" type="xs:integer"/>
                          <xs:element name="USED_MB" type="xs:integer"/>
                        </xs:all>
                      </xs:complexType>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="VMS">
      <xs:complexType>
        <xs:sequence>
```

```
                <xs:element name="ID" type="xs:integer" minOccurs="0" maxOccurs="unbounded"/>
              </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="TEMPLATE" type="xs:anyType"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
  targetNamespace="http://opennebula.org/XMLSchema" xmlns="http://opennebula.org/XMLSchema">
  <xs:include schemaLocation="host.xsd"/>
  <xs:element name="HOST_POOL">
    <xs:complexType>
      <xs:sequence maxOccurs="1" minOccurs="1">
        <xs:element ref="HOST" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Schemas for Image

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://opennebula.org/XMLSchema" elemer
  <xs:element name="IMAGE">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ID" type="xs:integer"/>
        <xs:element name="UID" type="xs:integer"/>
        <xs:element name="GID" type="xs:integer"/>
        <xs:element name="UNAME" type="xs:string"/>
        <xs:element name="GNAME" type="xs:string"/>
        <xs:element name="NAME" type="xs:string"/>
        <xs:element name="PERMISSIONS" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="OWNER_U" type="xs:integer"/>
              <xs:element name="OWNER_M" type="xs:integer"/>
              <xs:element name="OWNER_A" type="xs:integer"/>
              <xs:element name="GROUP_U" type="xs:integer"/>
              <xs:element name="GROUP_M" type="xs:integer"/>
              <xs:element name="GROUP_A" type="xs:integer"/>
              <xs:element name="OTHER_U" type="xs:integer"/>
              <xs:element name="OTHER_M" type="xs:integer"/>
              <xs:element name="OTHER_A" type="xs:integer"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="TYPE" type="xs:integer"/>
        <xs:element name="DISK_TYPE" type="xs:integer"/>
        <xs:element name="PERSISTENT" type="xs:integer"/>
        <xs:element name="REGTIME" type="xs:integer"/>
        <xs:element name="SOURCE" type="xs:string"/>
        <xs:element name="PATH" type="xs:string"/>
```

```
        <xs:element name="FSTYPE" type="xs:string"/>
        <xs:element name="SIZE" type="xs:integer"/>

        <!-- STATE values,
          INIT      = 0, Initialization state
          READY     = 1, Image ready to use
          USED      = 2, Image in use
          DISABLED  = 3, Image can not be instantiated by a VM
          LOCKED    = 4, FS operation for the Image in process
          ERROR     = 5, Error state the operation FAILED
          CLONE     = 6, Image is being cloned
          DELETE    = 7, DS is deleting the image
          USED_PERS = 8, Image is in use and persistent
        -->
        <xs:element name="STATE" type="xs:integer"/>
        <xs:element name="RUNNING_VMS" type="xs:integer"/>
        <xs:element name="CLONING_OPS" type="xs:integer"/>
        <xs:element name="CLONING_ID" type="xs:integer"/>
        <xs:element name="DATASTORE_ID" type="xs:integer"/>
        <xs:element name="DATASTORE" type="xs:string"/>
        <xs:element name="VMS">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="ID" type="xs:integer" minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
              </xs:complexType>
        </xs:element>
        <xs:element name="CLONES">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="ID" type="xs:integer" minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
              </xs:complexType>
        </xs:element>
        <xs:element name="TEMPLATE" type="xs:anyType"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
  targetNamespace="http://opennebula.org/XMLSchema" xmlns="http://opennebula.org/XMLSchema">
  <xs:include schemaLocation="image.xsd"/>
  <xs:element name="IMAGE_POOL">
    <xs:complexType>
      <xs:sequence maxOccurs="1" minOccurs="1">
        <xs:element ref="IMAGE" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Schemas for User

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
  targetNamespace="http://opennebula.org/XMLSchema" xmlns="http://opennebula.org/XMLSchema">
  <xs:element name="USER">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ID" type="xs:integer"/>
        <xs:element name="GID" type="xs:integer"/>
        <xs:element name="GROUPS">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="ID" type="xs:integer" minOccurs="1" maxOccurs="unbounded"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="GNAME" type="xs:string"/>
        <xs:element name="NAME" type="xs:string"/>
        <xs:element name="PASSWORD" type="xs:string"/>
        <xs:element name="AUTH_DRIVER" type="xs:string"/>
        <xs:element name="ENABLED" type="xs:integer"/>
        <xs:element name="TEMPLATE" type="xs:anyType"/>
        <xs:element name="DATASTORE_QUOTA" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="DATASTORE" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="ID" type="xs:string"/>
                  <xs:element name="IMAGES" type="xs:string"/>
                  <xs:element name="IMAGES_USED" type="xs:string"/>
                  <xs:element name="SIZE" type="xs:string"/>
                  <xs:element name="SIZE_USED" type="xs:string"/>
                </xs:sequence>
              </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="NETWORK_QUOTA" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="NETWORK" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="ID" type="xs:string"/>
                  <xs:element name="LEASES" type="xs:string"/>
                  <xs:element name="LEASES_USED" type="xs:string"/>
                </xs:sequence>
              </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="VM_QUOTA" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
```

```
        <xs:element name="VM" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="CPU" type="xs:string"/>
              <xs:element name="CPU_USED" type="xs:string"/>
              <xs:element name="MEMORY" type="xs:string"/>
              <xs:element name="MEMORY_USED" type="xs:string"/>
              <xs:element name="VMS" type="xs:string"/>
              <xs:element name="VMS_USED" type="xs:string"/>
              <xs:element name="VOLATILE_SIZE" type="xs:string"/>
              <xs:element name="VOLATILE_SIZE_USED" type="xs:string"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="IMAGE_QUOTA" minOccurs="0" maxOccurs="1">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="IMAGE" minOccurs="0" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="ID" type="xs:string"/>
              <xs:element name="RVMS" type="xs:string"/>
              <xs:element name="RVMS_USED" type="xs:string"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="DEFAULT_USER_QUOTAS">
    <xs:complexType>
      <xs:sequence>
          <xs:element name="DATASTORE_QUOTA" minOccurs="0" maxOccurs="1">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="DATASTORE" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="ID" type="xs:string"/>
                    <xs:element name="IMAGES" type="xs:string"/>
                    <xs:element name="IMAGES_USED" type="xs:string"/>
                    <xs:element name="SIZE" type="xs:string"/>
                    <xs:element name="SIZE_USED" type="xs:string"/>
                  </xs:sequence>
                </xs:complexType>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="NETWORK_QUOTA" minOccurs="0" maxOccurs="1">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="NETWORK" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:sequence>
```

```
                        <xs:element name="ID" type="xs:string"/>
                        <xs:element name="LEASES" type="xs:string"/>
                        <xs:element name="LEASES_USED" type="xs:string"/>
                      </xs:sequence>
                    </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="VM_QUOTA" minOccurs="0" maxOccurs="1">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="VM" minOccurs="0" maxOccurs="1">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="CPU" type="xs:string"/>
                          <xs:element name="CPU_USED" type="xs:string"/>
                          <xs:element name="MEMORY" type="xs:string"/>
                          <xs:element name="MEMORY_USED" type="xs:string"/>
                          <xs:element name="VMS" type="xs:string"/>
                          <xs:element name="VMS_USED" type="xs:string"/>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="IMAGE_QUOTA" minOccurs="0" maxOccurs="1">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="IMAGE" minOccurs="0" maxOccurs="unbounded">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="ID" type="xs:string"/>
                          <xs:element name="RVMS" type="xs:string"/>
                          <xs:element name="RVMS_USED" type="xs:string"/>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
  targetNamespace="http://opennebula.org/XMLSchema" xmlns="http://opennebula.org/XMLSchema">
  <xs:include schemaLocation="user.xsd"/>
  <xs:element name="USER_POOL">
    <xs:complexType>
      <xs:sequence maxOccurs="1" minOccurs="1">
        <xs:element name="USER" maxOccurs="unbounded" minOccurs="0">
```

```
<xs:complexType>
  <xs:sequence>
    <xs:element name="ID" type="xs:integer"/>
    <xs:element name="GID" type="xs:integer"/>
    <xs:element name="GROUPS">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="ID" type="xs:integer" minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="GNAME" type="xs:string"/>
    <xs:element name="NAME" type="xs:string"/>
    <xs:element name="PASSWORD" type="xs:string"/>
    <xs:element name="AUTH_DRIVER" type="xs:string"/>
    <xs:element name="ENABLED" type="xs:integer"/>
    <xs:element name="TEMPLATE" type="xs:anyType"/>
    <xs:element name="DATASTORE_QUOTA" minOccurs="0" maxOccurs="1">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="DATASTORE" minOccurs="0" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="ID" type="xs:string"/>
              <xs:element name="IMAGES" type="xs:string"/>
              <xs:element name="IMAGES_USED" type="xs:string"/>
              <xs:element name="SIZE" type="xs:string"/>
              <xs:element name="SIZE_USED" type="xs:string"/>
            </xs:sequence>
          </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="NETWORK_QUOTA" minOccurs="0" maxOccurs="1">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="NETWORK" minOccurs="0" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="ID" type="xs:string"/>
              <xs:element name="LEASES" type="xs:string"/>
              <xs:element name="LEASES_USED" type="xs:string"/>
            </xs:sequence>
          </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="VM_QUOTA" minOccurs="0" maxOccurs="1">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="VM" minOccurs="0" maxOccurs="1">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="CPU" type="xs:string"/>
                <xs:element name="CPU_USED" type="xs:string"/>
                <xs:element name="MEMORY" type="xs:string"/>
```

```
                    <xs:element name="MEMORY_USED" type="xs:string"/>
                    <xs:element name="VMS" type="xs:string"/>
                    <xs:element name="VMS_USED" type="xs:string"/>
                    <xs:element name="VOLATILE_SIZE" type="xs:string"/>
                    <xs:element name="VOLATILE_SIZE_USED" type="xs:string"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="IMAGE_QUOTA" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="IMAGE" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="ID" type="xs:string"/>
                    <xs:element name="RVMS" type="xs:string"/>
                    <xs:element name="RVMS_USED" type="xs:string"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="DEFAULT_USER_QUOTAS">
    <xs:complexType>
      <xs:sequence>
          <xs:element name="DATASTORE_QUOTA" minOccurs="0" maxOccurs="1">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="DATASTORE" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="ID" type="xs:string"/>
                    <xs:element name="IMAGES" type="xs:string"/>
                    <xs:element name="IMAGES_USED" type="xs:string"/>
                    <xs:element name="SIZE" type="xs:string"/>
                    <xs:element name="SIZE_USED" type="xs:string"/>
                  </xs:sequence>
                </xs:complexType>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="NETWORK_QUOTA" minOccurs="0" maxOccurs="1">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="NETWORK" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="ID" type="xs:string"/>
                    <xs:element name="LEASES" type="xs:string"/>
                    <xs:element name="LEASES_USED" type="xs:string"/>
```

```
                    </xs:sequence>
                  </xs:complexType>
                  </xs:element>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
            <xs:element name="VM_QUOTA" minOccurs="0" maxOccurs="1">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="VM" minOccurs="0" maxOccurs="1">
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element name="CPU" type="xs:string"/>
                        <xs:element name="CPU_USED" type="xs:string"/>
                        <xs:element name="MEMORY" type="xs:string"/>
                        <xs:element name="MEMORY_USED" type="xs:string"/>
                        <xs:element name="VMS" type="xs:string"/>
                        <xs:element name="VMS_USED" type="xs:string"/>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
            <xs:element name="IMAGE_QUOTA" minOccurs="0" maxOccurs="1">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="IMAGE" minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element name="ID" type="xs:string"/>
                        <xs:element name="RVMS" type="xs:string"/>
                        <xs:element name="RVMS_USED" type="xs:string"/>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Schemas for Virtual Machine

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
  targetNamespace="http://opennebula.org/XMLSchema" xmlns="http://opennebula.org/XMLSchema">
  <xs:element name="VM">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ID" type="xs:integer"/>
        <xs:element name="UID" type="xs:integer"/>
```

```
<xs:element name="GID" type="xs:integer"/>
<xs:element name="UNAME" type="xs:string"/>
<xs:element name="GNAME" type="xs:string"/>
<xs:element name="NAME" type="xs:string"/>
<xs:element name="PERMISSIONS" minOccurs="0" maxOccurs="1">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="OWNER_U" type="xs:integer"/>
      <xs:element name="OWNER_M" type="xs:integer"/>
      <xs:element name="OWNER_A" type="xs:integer"/>
      <xs:element name="GROUP_U" type="xs:integer"/>
      <xs:element name="GROUP_M" type="xs:integer"/>
      <xs:element name="GROUP_A" type="xs:integer"/>
      <xs:element name="OTHER_U" type="xs:integer"/>
      <xs:element name="OTHER_M" type="xs:integer"/>
      <xs:element name="OTHER_A" type="xs:integer"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="LAST_POLL" type="xs:integer"/>

<!-- STATE values,
see http://opennebula.org/_media/documentation:rel3.6:states-complete.png

  INIT      = 0
  PENDING   = 1
  HOLD      = 2
  ACTIVE    = 3 In this state, the Life Cycle Manager state is relevant
  STOPPED   = 4
  SUSPENDED = 5
  DONE      = 6
  FAILED    = 7
  POWEROFF  = 8
  UNDEPLOYED = 9
-->
<xs:element name="STATE" type="xs:integer"/>

<!-- LCM_STATE values, this sub-state is relevant only when STATE is
     ACTIVE (4)

  LCM_INIT            = 0,
  PROLOG              = 1,
  BOOT                = 2,
  RUNNING             = 3,
  MIGRATE             = 4,
  SAVE_STOP           = 5,
  SAVE_SUSPEND        = 6,
  SAVE_MIGRATE        = 7,
  PROLOG_MIGRATE      = 8,
  PROLOG_RESUME       = 9,
  EPILOG_STOP         = 10,
  EPILOG              = 11,
  SHUTDOWN            = 12,
  CANCEL              = 13,
  FAILURE             = 14,
  CLEANUP_RESUBMIT    = 15,
  UNKNOWN             = 16,
  HOTPLUG             = 17,
```

```
  SHUTDOWN_POWEROFF   = 18,
  BOOT_UNKNOWN        = 19,
  BOOT_POWEROFF       = 20,
  BOOT_SUSPENDED      = 21,
  BOOT_STOPPED        = 22,
  CLEANUP_DELETE      = 23,
  HOTPLUG_SNAPSHOT    = 24,
  HOTPLUG_NIC         = 25,
  HOTPLUG_SAVEAS           = 26,
  HOTPLUG_SAVEAS_POWEROFF  = 27,
  HOTPLUG_SAVEAS_SUSPENDED = 28,
  SHUTDOWN_UNDEPLOY   = 29,
  EPILOG_UNDEPLOY     = 30,
  PROLOG_UNDEPLOY     = 31,
  BOOT_UNDEPLOY       = 32
-->
<xs:element name="LCM_STATE" type="xs:integer"/>
<xs:element name="RESCHED" type="xs:integer"/>
<xs:element name="STIME" type="xs:integer"/>
<xs:element name="ETIME" type="xs:integer"/>
<xs:element name="DEPLOY_ID" type="xs:string"/>

<!-- MEMORY consumption in kilobytes -->
<xs:element name="MEMORY" type="xs:integer"/>

<!-- Percentage of 1 CPU consumed (two fully consumed cpu is 200) -->
<xs:element name="CPU" type="xs:integer"/>

<!-- NET_TX: Sent bytes to the network -->
<xs:element name="NET_TX" type="xs:integer"/>

<!-- NET_RX: Received bytes from the network -->
<xs:element name="NET_RX" type="xs:integer"/>
<xs:element name="TEMPLATE" type="xs:anyType"/>
<xs:element name="USER_TEMPLATE" type="xs:anyType"/>
<xs:element name="HISTORY_RECORDS">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="HISTORY" maxOccurs="unbounded" minOccurs="0">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="OID" type="xs:integer"/>
            <xs:element name="SEQ" type="xs:integer"/>
            <xs:element name="HOSTNAME" type="xs:string"/>
            <xs:element name="HID" type="xs:integer"/>
            <xs:element name="CID" type="xs:integer"/>
            <xs:element name="STIME" type="xs:integer"/>
            <xs:element name="ETIME" type="xs:integer"/>
            <xs:element name="VMMMAD" type="xs:string"/>
            <xs:element name="VNMMAD" type="xs:string"/>
            <xs:element name="TMMAD" type="xs:string"/>
            <xs:element name="DS_LOCATION" type="xs:string"/>
            <xs:element name="DS_ID" type="xs:integer"/>
            <xs:element name="PSTIME" type="xs:integer"/>
            <xs:element name="PETIME" type="xs:integer"/>
            <xs:element name="RSTIME" type="xs:integer"/>
            <xs:element name="RETIME" type="xs:integer"/>
            <xs:element name="ESTIME" type="xs:integer"/>
```

```
                    <xs:element name="EETIME" type="xs:integer"/>

                    <!-- REASON values:
                      NONE  = 0 History record is not closed yet
                      ERROR = 1 History record was closed because of an error
                      USER  = 2 History record was closed because of a user action
                    -->
                    <xs:element name="REASON" type="xs:integer"/>

                    <!-- ACTION values:
                      NONE_ACTION             = 0
                      MIGRATE_ACTION          = 1
                      LIVE_MIGRATE_ACTION     = 2
                      SHUTDOWN_ACTION         = 3
                      SHUTDOWN_HARD_ACTION    = 4
                      UNDEPLOY_ACTION         = 5
                      UNDEPLOY_HARD_ACTION    = 6
                      HOLD_ACTION             = 7
                      RELEASE_ACTION          = 8
                      STOP_ACTION             = 9
                      SUSPEND_ACTION          = 10
                      RESUME_ACTION           = 11
                      BOOT_ACTION             = 12
                      DELETE_ACTION           = 13
                      DELETE_RECREATE_ACTION  = 14
                      REBOOT_ACTION           = 15
                      REBOOT_HARD_ACTION      = 16
                      RESCHED_ACTION          = 17
                      UNRESCHED_ACTION        = 18
                      POWEROFF_ACTION         = 19
                      POWEROFF_HARD_ACTION    = 20
                    -->
                    <xs:element name="ACTION" type="xs:integer"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="unqualified"
    targetNamespace="http://opennebula.org/XMLSchema" xmlns="http://opennebula.org/XMLSchema">
    <xs:include schemaLocation="vm.xsd"/>
    <xs:element name="VM_POOL">
        <xs:complexType>
            <xs:sequence maxOccurs="1" minOccurs="1">
                <xs:element ref="VM" maxOccurs="unbounded" minOccurs="0"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Schemas for Virtual Machine Template

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://opennebula.org/XMLSchema" elemer
  <xs:element name="VMTEMPLATE">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ID" type="xs:integer"/>
        <xs:element name="UID" type="xs:integer"/>
        <xs:element name="GID" type="xs:integer"/>
        <xs:element name="UNAME" type="xs:string"/>
        <xs:element name="GNAME" type="xs:string"/>
        <xs:element name="NAME" type="xs:string"/>
        <xs:element name="PERMISSIONS" minOccurs="1" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="OWNER_U" type="xs:integer"/>
              <xs:element name="OWNER_M" type="xs:integer"/>
              <xs:element name="OWNER_A" type="xs:integer"/>
              <xs:element name="GROUP_U" type="xs:integer"/>
              <xs:element name="GROUP_M" type="xs:integer"/>
              <xs:element name="GROUP_A" type="xs:integer"/>
              <xs:element name="OTHER_U" type="xs:integer"/>
              <xs:element name="OTHER_M" type="xs:integer"/>
              <xs:element name="OTHER_A" type="xs:integer"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="REGTIME" type="xs:integer"/>
        <xs:element name="TEMPLATE" type="xs:anyType"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
  targetNamespace="http://opennebula.org/XMLSchema" xmlns="http://opennebula.org/XMLSchema">
  <xs:include schemaLocation="vmtemplate.xsd"/>
  <xs:element name="VMTEMPLATE_POOL">
    <xs:complexType>
      <xs:sequence maxOccurs="1" minOccurs="1">
        <xs:element ref="VMTEMPLATE" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Schemas for Virtual Network

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
  targetNamespace="http://opennebula.org/XMLSchema" xmlns="http://opennebula.org/XMLSchema">
  <xs:element name="VNET">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ID" type="xs:integer"/>
```

```
<xs:element name="UID" type="xs:integer"/>
<xs:element name="GID" type="xs:integer"/>
<xs:element name="UNAME" type="xs:string"/>
<xs:element name="GNAME" type="xs:string"/>
<xs:element name="NAME" type="xs:string"/>
<xs:element name="PERMISSIONS" minOccurs="0" maxOccurs="1">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="OWNER_U" type="xs:integer"/>
      <xs:element name="OWNER_M" type="xs:integer"/>
      <xs:element name="OWNER_A" type="xs:integer"/>
      <xs:element name="GROUP_U" type="xs:integer"/>
      <xs:element name="GROUP_M" type="xs:integer"/>
      <xs:element name="GROUP_A" type="xs:integer"/>
      <xs:element name="OTHER_U" type="xs:integer"/>
      <xs:element name="OTHER_M" type="xs:integer"/>
      <xs:element name="OTHER_A" type="xs:integer"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="CLUSTER_ID" type="xs:integer"/>
<xs:element name="CLUSTER" type="xs:string"/>
<xs:element name="TYPE" type="xs:integer"/>
<xs:element name="BRIDGE" type="xs:string"/>
<xs:element name="VLAN" type="xs:integer"/>
<xs:element name="PHYDEV" type="xs:string"/>
<xs:element name="VLAN_ID" type="xs:string"/>
<xs:element name="GLOBAL_PREFIX" type="xs:string"/>
<xs:element name="SITE_PREFIX" type="xs:string"/>
<xs:element name="RANGE" minOccurs="0" maxOccurs="1">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="IP_START" type="xs:string"/>
      <xs:element name="IP_END" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="TOTAL_LEASES" type="xs:integer"/>
<xs:element name="TEMPLATE" type="xs:anyType"/>
<xs:element name="LEASES" minOccurs="0" maxOccurs="1">
  <xs:complexType>
    <xs:sequence minOccurs="0">
      <xs:element name="LEASE" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="MAC" type="xs:string"/>
            <xs:element name="IP" type="xs:string"/>
            <xs:element name="IP6_LINK" type="xs:string"/>
            <xs:element name="IP6_SITE" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="IP6_GLOBAL" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="USED" type="xs:integer"/>
            <xs:element name="VID" type="xs:integer"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
  targetNamespace="http://opennebula.org/XMLSchema" xmlns="http://opennebula.org/XMLSchema">
  <xs:include schemaLocation="vnet.xsd"/>
  <xs:element name="VNET_POOL">
    <xs:complexType>
      <xs:sequence maxOccurs="1" minOccurs="1">
        <xs:element ref="VNET" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Schemas for Accounting

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
  targetNamespace="http://opennebula.org/XMLSchema" xmlns="http://opennebula.org/XMLSchema">

  <xs:element name="HISTORY_RECORDS">
    <xs:complexType>
      <xs:sequence maxOccurs="1" minOccurs="1">
        <xs:element ref="HISTORY" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="HISTORY">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="OID" type="xs:integer"/>
        <xs:element name="SEQ" type="xs:integer"/>
        <xs:element name="HOSTNAME" type="xs:string"/>
        <xs:element name="HID" type="xs:integer"/>
        <xs:element name="CID" type="xs:integer"/>
        <xs:element name="STIME" type="xs:integer"/>
        <xs:element name="ETIME" type="xs:integer"/>
        <xs:element name="VMMMAD" type="xs:string"/>
        <xs:element name="VNMMAD" type="xs:string"/>
        <xs:element name="TMMAD" type="xs:string"/>
        <xs:element name="DS_LOCATION" type="xs:string"/>
        <xs:element name="DS_ID" type="xs:integer"/>
        <xs:element name="PSTIME" type="xs:integer"/>
        <xs:element name="PETIME" type="xs:integer"/>
        <xs:element name="RSTIME" type="xs:integer"/>
        <xs:element name="RETIME" type="xs:integer"/>
        <xs:element name="ESTIME" type="xs:integer"/>
        <xs:element name="EETIME" type="xs:integer"/>

        <!-- REASON values:
          NONE  = 0 History record is not closed yet
          ERROR = 1 History record was closed because of an error
```

```
  USER  = 2 History record was closed because of a user action
-->
<xs:element name="REASON" type="xs:integer"/>

<!-- ACTION values:
  NONE_ACTION             = 0
  MIGRATE_ACTION          = 1
  LIVE_MIGRATE_ACTION     = 2
  SHUTDOWN_ACTION         = 3
  SHUTDOWN_HARD_ACTION    = 4
  UNDEPLOY_ACTION         = 5
  UNDEPLOY_HARD_ACTION    = 6
  HOLD_ACTION             = 7
  RELEASE_ACTION          = 8
  STOP_ACTION             = 9
  SUSPEND_ACTION          = 10
  RESUME_ACTION           = 11
  BOOT_ACTION             = 12
  DELETE_ACTION           = 13
  DELETE_RECREATE_ACTION  = 14
  REBOOT_ACTION           = 15
  REBOOT_HARD_ACTION      = 16
  RESCHED_ACTION          = 17
  UNRESCHED_ACTION        = 18
  POWEROFF_ACTION         = 19
  POWEROFF_HARD_ACTION    = 20
-->
<xs:element name="ACTION" type="xs:integer"/>

<xs:element name="VM">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="ID" type="xs:integer"/>
      <xs:element name="UID" type="xs:integer"/>
      <xs:element name="GID" type="xs:integer"/>
      <xs:element name="UNAME" type="xs:string"/>
      <xs:element name="GNAME" type="xs:string"/>
      <xs:element name="NAME" type="xs:string"/>
      <xs:element name="PERMISSIONS" minOccurs="0" maxOccurs="1">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="OWNER_U" type="xs:integer"/>
            <xs:element name="OWNER_M" type="xs:integer"/>
            <xs:element name="OWNER_A" type="xs:integer"/>
            <xs:element name="GROUP_U" type="xs:integer"/>
            <xs:element name="GROUP_M" type="xs:integer"/>
            <xs:element name="GROUP_A" type="xs:integer"/>
            <xs:element name="OTHER_U" type="xs:integer"/>
            <xs:element name="OTHER_M" type="xs:integer"/>
            <xs:element name="OTHER_A" type="xs:integer"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="LAST_POLL" type="xs:integer"/>

      <!-- STATE values,
      see http://opennebula.org/documentation:documentation:api#actions_for_virtual_machine_r
```

```
  INIT      = 0
  PENDING   = 1
  HOLD      = 2
  ACTIVE    = 3 In this state, the Life Cycle Manager state is relevant
  STOPPED   = 4
  SUSPENDED = 5
  DONE      = 6
  FAILED    = 7
  POWEROFF  = 8
  UNDEPLOYED = 9
-->
<xs:element name="STATE" type="xs:integer"/>

<!-- LCM_STATE values, this sub-state is relevant only when STATE is
     ACTIVE (4)

  LCM_INIT            = 0,
  PROLOG              = 1,
  BOOT                = 2,
  RUNNING             = 3,
  MIGRATE             = 4,
  SAVE_STOP           = 5,
  SAVE_SUSPEND        = 6,
  SAVE_MIGRATE        = 7,
  PROLOG_MIGRATE      = 8,
  PROLOG_RESUME       = 9,
  EPILOG_STOP         = 10,
  EPILOG              = 11,
  SHUTDOWN            = 12,
  CANCEL              = 13,
  FAILURE             = 14,
  CLEANUP_RESUBMIT    = 15,
  UNKNOWN             = 16,
  HOTPLUG             = 17,
  SHUTDOWN_POWEROFF   = 18,
  BOOT_UNKNOWN        = 19,
  BOOT_POWEROFF       = 20,
  BOOT_SUSPENDED      = 21,
  BOOT_STOPPED        = 22,
  CLEANUP_DELETE      = 23,
  HOTPLUG_SNAPSHOT    = 24,
  HOTPLUG_NIC         = 25,
  HOTPLUG_SAVEAS           = 26,
  HOTPLUG_SAVEAS_POWEROFF  = 27,
  HOTPLUG_SAVEAS_SUSPENDED = 28,
  SHUTDOWN_UNDEPLOY   = 29,
  EPILOG_UNDEPLOY     = 30,
  PROLOG_UNDEPLOY     = 31,
  BOOT_UNDEPLOY       = 32
-->
<xs:element name="LCM_STATE" type="xs:integer"/>
<xs:element name="RESCHED" type="xs:integer"/>
<xs:element name="STIME" type="xs:integer"/>
<xs:element name="ETIME" type="xs:integer"/>
<xs:element name="DEPLOY_ID" type="xs:string"/>

<!-- MEMORY consumption in kilobytes -->
<xs:element name="MEMORY" type="xs:integer"/>
```

```
            <!-- Percentage of 1 CPU consumed (two fully consumed cpu is 200) -->
            <xs:element name="CPU" type="xs:integer"/>

            <!-- NET_TX: Sent bytes to the network -->
            <xs:element name="NET_TX" type="xs:integer"/>

            <!-- NET_RX: Received bytes from the network -->
            <xs:element name="NET_RX" type="xs:integer"/>
            <xs:element name="TEMPLATE" type="xs:anyType"/>
            <xs:element name="USER_TEMPLATE" type="xs:anyType"/>
            <xs:element name="HISTORY_RECORDS">
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

3.2 Ruby OpenNebula Cloud API

This page contains the OpenNebula Cloud API Specification for Ruby. It has been designed as a wrapper for the *XML-RPC methods*, with some basic helpers. This means that you should be familiar with the XML-RPC API and the XML formats returned by the OpenNebula core. As stated in the *XML-RPC documentation*, you can download the *XML Schemas (XSD) here*.

3.2.1 API Documentation

You can consult the doc online.

3.2.2 Usage

You can use the Ruby OCA included in the OpenNebula distribution by adding the OpenNebula Ruby library path to the search path:

```
################################################################################
# Environment Configuration
################################################################################
ONE_LOCATION=ENV["ONE_LOCATION"]

if !ONE_LOCATION
    RUBY_LIB_LOCATION="/usr/lib/one/ruby"
else
    RUBY_LIB_LOCATION=ONE_LOCATION+"/lib/ruby"
end

$: << RUBY_LIB_LOCATION

################################################################################
# Required libraries
################################################################################
require 'opennebula'
```

3.2.3 Code Sample: Shutdown all the VMs of the Pool

This is a small code snippet. As you can see, the code flow would be as follows:

- Create a new Client, setting up the authorization string. You only need to create it once.
- Get the VirtualMachine pool that contains the VirtualMachines owned by this User.
- You can perform "actions" over these objects right away, like `myVNet.delete();`. In this example all the VirtualMachines will be shut down.

```
#!/usr/bin/env ruby

################################################################################
# Environment Configuration
################################################################################
ONE_LOCATION=ENV["ONE_LOCATION"]

if !ONE_LOCATION
    RUBY_LIB_LOCATION="/usr/lib/one/ruby"
else
    RUBY_LIB_LOCATION=ONE_LOCATION+"/lib/ruby"
end

$: << RUBY_LIB_LOCATION

################################################################################
# Required libraries
################################################################################
require 'opennebula'

include OpenNebula

# OpenNebula credentials
CREDENTIALS = "oneuser:onepass"
# XML_RPC endpoint where OpenNebula is listening
ENDPOINT    = "http://localhost:2633/RPC2"

client = Client.new(CREDENTIALS, ENDPOINT)

vm_pool = VirtualMachinePool.new(client, -1)

rc = vm_pool.info
if OpenNebula.is_error?(rc)
     puts rc.message
     exit -1
end

vm_pool.each do |vm|
     rc = vm.shutdown
     if OpenNebula.is_error?(rc)
          puts "Virtual Machine #{vm.id}: #{rc.message}"
     else
          puts "Virtual Machine #{vm.id}: Shutting down"
     end
end

exit 0
```

3.2.4 Code Sample: Create a new VirtualNetwork

```
#!/usr/bin/env ruby

################################################################################
# Environment Configuration
################################################################################
ONE_LOCATION=ENV["ONE_LOCATION"]

if !ONE_LOCATION
    RUBY_LIB_LOCATION="/usr/lib/one/ruby"
else
    RUBY_LIB_LOCATION=ONE_LOCATION+"/lib/ruby"
end

$: << RUBY_LIB_LOCATION

################################################################################
# Required libraries
################################################################################
require 'opennebula'

include OpenNebula

# OpenNebula credentials
CREDENTIALS = "oneuser:onepass"
# XML_RPC endpoint where OpenNebula is listening
ENDPOINT    = "http://localhost:2633/RPC2"

client = Client.new(CREDENTIALS, ENDPOINT)

template = <<-EOT
NAME    = "Red LAN"
TYPE    = RANGED

# Now we'll use the host private network (physical)
BRIDGE  = vbr0

NETWORK_SIZE    = C
NETWORK_ADDRESS = 192.168.0.0

# Custom Attributes to be used in Context
GATEWAY = 192.168.0.1
DNS     = 192.168.0.1

LOAD_BALANCER = 192.168.0.3
EOT

xml = OpenNebula::VirtualNetwork.build_xml
vn  = OpenNebula::VirtualNetwork.new(xml, @client)

rc = vn.allocate(template)
if OpenNebula.is_error?(rc)
    exit -1, rc.message
else
    puts "ID: #{vn.id.to_s}"
end
```

```
puts "Before info:"
puts vn.to_xml

puts

vn.info

puts "After info:"
puts vn.to_xml
```

3.3 Java OpenNebula Cloud API

This page contains the OpenNebula Cloud API Specification for Java. It has been designed as a wrapper for the *XML-RPC methods*, with some basic helpers. This means that you should be familiar with the XML-RPC API and the XML formats returned by the OpenNebula core. As stated in the *XML-RPC documentation*, you can download the *XML Schemas (XSD) here*.

3.3.1 Download

The Java OCA is part of the OpenNebula core distribution. If you installed from the Debian, Ubuntu or CentOS packages it should be already installed in `/usr/share/java/org.opennebula.client.jar`. You also can download the `.jar` file compiled using Java OpenJDK 1.7, the required libraries, and the javadoc packaged in a tar.gz file following this link.

You can also consult the javadoc online.

3.3.2 Usage

To use the OpenNebula Cloud API for Java in your Java project, you have to add to the classpath the org.opennebula.client.jar file and the xml-rpc libraries located in the lib directory.

3.3.3 Code Sample

This is a small code snippet. As you can see, the code flow would be as follows:

- Create a `org.opennebula.client.Client` object, setting up the authorization string and the endpoint. You only need to create it once.
- Create a pool (e.g. `HostPool`) or element (e.g. `VirtualNetwork`) object.
- You can perform "actions" over these objects right away, like `myVNet.delete();`
- If you want to query any information (like what objects the pool contains, or one of the element attributes), you have to issue an `info()` call before, so the object retrieves the data from OpenNebula.

For more complete examples, please check the `src/oca/java/share/examples` directory included. You may be also interested in the java files included in `src/oca/java/test`.

```
// First of all, a Client object has to be created.
// Here the client will try to connect to OpenNebula using the default
// options: the auth. file will be assumed to be at $ONE_AUTH, and the
// endpoint will be set to the environment variable $ONE_XMLRPC.
Client oneClient;
```

```
try
{
    oneClient = new Client();

    // This VM template is a valid one, but it will probably fail to run
    // if we try to deploy it; the path for the image is unlikely to
    // exist.
    String vmTemplate =
          "NAME     = vm_from_java    CPU = 0.1    MEMORY = 64\n"
        + "DISK     = [\n"
        + "\tsource   = \"/home/user/vmachines/ttylinux/ttylinux.img\",\n"
        + "\ttarget   = \"hda\",\n"
        + "\treadonly = \"no\" ]\n"
        + "FEATURES = [ acpi=\"no\" ]";

    System.out.print("Trying to allocate the virtual machine... ");
    OneResponse rc = VirtualMachine.allocate(oneClient, vmTemplate);

    if( rc.isError() )
    {
        System.out.println( "failed!");
        throw new Exception( rc.getErrorMessage() );
    }

    // The response message is the new VM's ID
    int newVMID = Integer.parseInt(rc.getMessage());
    System.out.println("ok, ID " + newVMID + ".");

    // We can create a representation for the new VM, using the returned
    // VM-ID
    VirtualMachine vm = new VirtualMachine(newVMID, oneClient);

    // Let's hold the VM, so the scheduler won't try to deploy it
    System.out.print("Trying to hold the new VM... ");
    rc = vm.hold();

    if(rc.isError())
    {
        System.out.println("failed!");
        throw new Exception( rc.getErrorMessage() );
    }

    // And now we can request its information.
    rc = vm.info();

    if(rc.isError())
        throw new Exception( rc.getErrorMessage() );

    System.out.println();
    System.out.println(
            "This is the information OpenNebula stores for the new VM:");
    System.out.println(rc.getMessage() + "\n");

    // This VirtualMachine object has some helpers, so we can access its
    // attributes easily (remember to load the data first using the info
    // method).
    System.out.println("The new VM " +
            vm.getName() + " has status: " + vm.status());
```

```
    // And we can also use xpath expressions
    System.out.println("The path of the disk is");
    System.out.println( "\t" + vm.xpath("template/disk/source") );

    // We have also some useful helpers for the actions you can perform
    // on a virtual machine, like cancel or finalize:

    rc = vm.finalizeVM();
    System.out.println("\nTrying to finalize (delete) the VM " +
                       vm.getId() + "...");

}
catch (Exception e)
{
    System.out.println(e.getMessage());
}
```

3.3.4 Compilation

To compile the Java OCA, untar the OpenNebula source, `cd` to the java directory and use the build script:

```
$ cd src/oca/java
$ ./build.sh -d
Compiling java files into class files...
Packaging class files in a jar...
Generating javadocs...
```

This command will compile and package the code in `jar/org.opennebula.client.jar`, and the javadoc will be created in `share/doc/`.

You might want to copy the .jar files to a more convenient directory. You could use /usr/lib/one/java/

```
$ sudo mkdir /usr/lib/one/java/
$ sudo cp jar/* lib/* /usr/lib/one/java/
```

3.4 Ruby OpenNebula Zone API

This page contains the OpenNebula Zone API (ZONA) Specification for Ruby. It has been designed as a wrapper for the OpenNebula Zone REST server, with some basic helpers. This means that you should be familiar with the XML-RPC API and the JSON formats returned by the OpenNebula Zone server.

3.4.1 API Documentation

You can consult the doc online.

3.4.2 Usage

You can use the Ruby ZONA included in the OpenNebula distribution by adding the OpenNebula Ruby library path to the search path:

```
################################################################################
# Environment Configuration
################################################################################
ONE_LOCATION=ENV["ONE_LOCATION"]

if !ONE_LOCATION
    RUBY_LIB_LOCATION="/usr/lib/one/ruby"
else
    RUBY_LIB_LOCATION=ONE_LOCATION+"/lib/ruby"
end

$: << RUBY_LIB_LOCATION

################################################################################
# Required libraries
################################################################################
require 'zona'
```

3.4.3 Code Sample

This is a small code snippet. As you can see, the code flow would be as follows:

```
#!/usr/bin/env ruby

################################################################################
# Environment Configuration
################################################################################
ONE_LOCATION=ENV["ONE_LOCATION"]

if !ONE_LOCATION
    RUBY_LIB_LOCATION="/usr/lib/one/ruby"
else
    RUBY_LIB_LOCATION=ONE_LOCATION+"/lib/ruby"
end

$: << RUBY_LIB_LOCATION

################################################################################
# Required libraries
################################################################################
require 'zona'

TDB
```

3.5 OneFlow Specification

3.5.1 Overview

The OpenNebula OneFlow API is a RESTful service to create, control and monitor multi-tier applications or services composed of interconnected Virtual Machines with deployment dependencies between them. Each group of Virtual Machines is deployed and managed as a single entity, and is completely integrated with the advanced OpenNebula user and group management. There are two kind of resources; services templates and services. All data is sent and received as JSON.

This guide is intended for developers. The OpenNebula distribution includes a *cli* to interact with OneFlow and it is also fully integrated in the *Sunstone GUI*

3.5.2 Authentication & Authorization

User authentication will be HTTP Basic access authentication. The credentials passed should be the User name and password.

```
$ curl -u "username:password" https://oneflow.server
```

3.5.3 Return Codes

The OneFlow API uses the following subset of HTTP Status codes:

- **200 OK** : The request has succeeded.
- **201 Created** : Request was successful and a new resource has being created
- **202 Accepted** : The request has been accepted for processing, but the processing has not been completed
- **204 No Content** : The request has been accepted for processing, but no info in the response
- **400 Bad Request** : Malformed syntax
- **401 Unauthorized** : Bad authentication
- **403 Forbidden** : Bad authorization
- **404 Not Found** : Resource not found
- **500 Internal Server Error** : The server encountered an unexpected condition which prevented it from fulfilling the request.
- **501 Not Implemented** : The functionality requested is not supported

```
> POST /service_template HTTP/1.1
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.14.0.0 zlib/1.2.3 libidn/1.1
> Host: onflow.server:2474
>
< HTTP/1.1 400 Bad Request
< Content-Type: text/html;charset=utf-8
< Content-Type:application/json;charset=utf-8
< Content-Length: 40
<
{
  "error": {
    "message": "Role 'worker' 'cardinality' must be greater than or equal to 'min_vms'"
  }
}
```

The methods specified below are described without taking into account **4xx** (can be inferred from authorization information in section above) and **5xx** errors (which are method independent). HTTP verbs not defined for a particular entity will return a **501 Not Implemented**.

3.5.4 Methods

Service

Method	URL	Meaning / Entity Body	Response
GET	`/service`	**List** the contents of the `SERVICE` collection.	**200 OK**: A JSON representation of the collection in the http body
GET	`/service/<id>`	**Show** the `SERVICE` resource identified by <id>	**200 OK**: A JSON representation of the collection in the http body
DELETE	`/service/<id>`	**Delete** the `SERVICE` resource identified by <id>	**201**:
POST	`/service/<id>`	**Perform** an action on the `SERVICE` resource identified by <id>. Available actions: shutdown, recover, chown, chgrp, chmod	**201**:
PUT	`/service/<id>/role/<name>`	**Update** the `ROLE` identified by <name> of the `SERVICE` resource identified by <id>. Currently the only attribute that can be updated is the cardinality.	**200 OK**:
POST	`/service/<id>/role_action/<name>`	**Perform** an action on all the Virtual Machines belonging to the `ROLE` identified by <name> of the `SERVICE` resource identified by <id>. Available actions: shutdown, shutdown-hard, undeploy, undeploy-hard, hold, release, stop, suspend, resume, boot, delete, delete-recreate, reboot, reboot-hard, poweroff, poweroff-hard, snapshot-create	**201**:

Service Template

Method	URL	Meaning / Entity Body	Response
GET	`/service_template`	**List** the contents of the `SERVICE_TEMPLATE` collection.	**200 OK**: A JSON representation of the collection in the http body
GET	`/service_template/<id>`	**Show** the `SERVICE_TEMPLATE` resource identified by <id>	**200 OK**: A JSON representation of the collection in the http body
DELETE	`/service_template/<id>`	**Delete** the `SERVICE_TEMPLATE` resource identified by <id>	**201**:
POST	`/service_template`	**Create** a new `SERVICE_TEMPLATE` resource.	**201 Created**: A JSON representation of the new `SERVICE_TEMPLATE` resource in the http body
PUT	`/service_template/<id>`	**Update** the `SERVICE_TEMPLATE` resource identified by <id>.	**200 OK**:
POST	`/service_template/<id>/action`	**Perform** an action on the `SERVICE_TEMPLATE` resource identified by <id>. Available actions: instantiate, chown, chgrp, chmod	**201**:

3.5.5 Resource Representation

Service Schema

A Service is defined with JSON syntax templates.

Attribute	Type	Mandatory	Description
name	string	No	Name of the Service
deployment	string	No	**Deployment strategy:** **none**: All roles are deployed at the same time **straight**: Each Role is deployed when all its parent Roles are running Defaults to none
shutdown_action	string	No	VM shutdown action: 'shutdown' or 'shutdown-hard'. If it is not set, the default set in oneflow-server.conf will be used
roles	array of Roles	Yes	Array of Roles, see below

Each Role is defined as:

Attribute	Type	Mandatory	Description
name	string	Yes	Role name
cardinality	integer	No	Number of VMs to deploy. Defaults to 1
vm_template	integer	Yes	OpenNebula VM Template ID. See the *OpenNebula documentation for VM Templates*
parents	array of string	No	Names of the roles that must be deployed before this one
shut-down_action	string	No	VM shutdown action: 'shutdown' or 'shutdown-hard'. If it is not set, the one set for the Service will be used
min_vms	integer	No (Yes for elasticity)	Minimum number of VMs for elasticity adjustments
max_vms	integer	No (Yes for elasticity)	Maximum number of VMs for elasticity adjustments
cooldown	integer	No	Cooldown period duration after a scale operation, in seconds. If it is not set, the default set in oneflow-server.conf will be used
elastic-ity_policies	array of Policies	No	Array of Elasticity Policies, see below
sched-uled_policies	array of Policies	No	Array of Scheduled Policies, see below

To define a elasticity policy:

Attribute	Type	Mandatory	Description
type	string	Yes	Type of adjustment. Values: CHANGE, CARDINALITY, PERCENTAGE_CHANGE
adjust	integer	Yes	Positive or negative adjustment. Its meaning depends on 'type'
min_adjust_step	integer	No	Optional parameter for PERCENTAGE_CHAGE adjustment type. If present, the policy will change the cardinality by at least the number of VMs set in this attribute.
expression	string	Yes	Expression to trigger the elasticity
period_number	integer	No	Number of periods that the expression must be true before the elasticity is triggered
period	integer	No	Duration, in seconds, of each period in period_duration
cooldown	integer	No	Cooldown period duration after a scale operation, in seconds. If it is not set, the one set for the Role will be used

And each scheduled policy is defined as:

Attribute	Type	Mandatory	Description
type	string	Yes	Type of adjustment. Values: CHANGE, CARDINALITY, PERCENTAGE_CHANGE
adjust	integer	Yes	Positive or negative adjustment. Its meaning depends on 'type'
min_adjust_step	integer	No	Optional parameter for PERCENTAGE_CHAGE adjustment type. If present, the policy will change the cardinality by at least the number of VMs set in this attribute.
recurrence	string	No	Time for recurring adjustements. Time is specified with the Unix cron syntax
start_time	string	No	Exact time for the adjustement

```
{
  :type => :object,
  :properties => {
    'name' => {
      :type => :string,
      :required => true
    },
    'deployment' => {
      :type => :string,
      :enum => %w{none straight},
      :default => 'none'
    },
    'shutdown_action' => {
      :type => :string,
      :enum => %w{shutdown shutdown-hard},
      :required => false
    },
    'roles' => {
      :type => :array,
      :items => ROLE_SCHEMA,
      :required => true
```

```
    }
  }
}
```

Role Schema

```
{
  :type => :object,
  :properties => {
    'name' => {
      :type => :string,
      :required => true
    },
    'cardinality' => {
      :type => :integer,
      :default => 1,
      :minimum => 0
    },
    'vm_template' => {
      :type => :integer,
      :required => true
    },
    'parents' => {
      :type => :array,
      :items => {
        :type => :string
      }
    },
    'shutdown_action' => {
      :type => :string,
      :enum => ['shutdown', 'shutdown-hard']},
      :required => false
    },
    'min_vms' => {
      :type => :integer,
      :required => false,
      :minimum => 0
    },
    'max_vms' => {
      :type => :integer,
      :required => false,
      :minimum => 0
    },
    'cooldown' => {
      :type => :integer,
      :required => false,
      :minimum => 0
    },
    'elasticity_policies' => {
      :type => :array,
      :items => {
        :type => :object,
        :properties => {
          'type' => {
            :type => :string,
            :enum => ['CHANGE', 'CARDINALITY', 'PERCENTAGE_CHANGE'],
```

```
          :required => true
        },
        'adjust' => {
          :type => :integer,
          :required => true
        },
        'min_adjust_step' => {
          :type => :integer,
          :required => false,
          :minimum => 1
        },
        'period_number' => {
          :type => :integer,
          :required => false,
          :minimum => 0
        },
        'period' => {
          :type => :integer,
          :required => false,
          :minimum => 0
        },
        'expression' => {
          :type => :string,
          :required => true
        },
        'cooldown' => {
          :type => :integer,
          :required => false,
          :minimum => 0
        }
      }
    }
  },
  'scheduled_policies' => {
    :type => :array,
    :items => {
      :type => :object,
      :properties => {
        'type' => {
          :type => :string,
          :enum => ['CHANGE', 'CARDINALITY', 'PERCENTAGE_CHANGE'],
          :required => true
        },
        'adjust' => {
          :type => :integer,
          :required => true
        },
        'min_adjust_step' => {
          :type => :integer,
          :required => false,
          :minimum => 1
        },
        'start_time' => {
          :type => :string,
          :required => false
        },
        'recurrence' => {
          :type => :string,
```

```
            :required => false
          }
        }
      }
    }
  }
}
```

Action Schema

```
{
  :type => :object,
  :properties => {
    'action' => {
      :type => :object,
      :properties => {
        'perform' => {
          :type => :string,
          :required => true
        },
        'params' => {
          :type => :object,
            :required => false
          }
        }
      }
    }
  }
}
```

3.5.6 Examples

Create a New Service Template

Method	URL	Meaning / Entity Body	Response
POST	/service_template	**Create** a new SERVICE_TEMPLATE resource.	**201 Created**: A JSON representation of the new SERVICE_TEMPLATE resource in the http body

```
curl http://127.0.0.1:2474/service_template -u 'oneadmin:password' -v --data '{
  "name":"web-application",
  "deployment":"straight",
  "roles":[
    {
      "name":"frontend",
      "cardinality":"1",
      "vm_template":"0",
      "shutdown_action":"shutdown",
      "min_vms":"1",
      "max_vms":"4",
      "cooldown":"30",
      "elasticity_policies":[
        {
          "type":"PERCENTAGE_CHANGE",
          "adjust":"20",
```

```
          "min_adjust_step":"1",
          "expression":"CUSTOM_ATT>40",
          "period":"3",
          "period_number":"30",
          "cooldown":"30"
        }
      ],
      "scheduled_policies":[
        {
          "type":"CHANGE",
          "adjust":"4",
          "recurrence":"0 2 1-10 * *"
        }
      ]
    },
    {
      "name":"worker",
      "cardinality":"2",
      "vm_template":"0",
      "shutdown_action":"shutdown",
      "parents":[
        "frontend"
      ],
      "min_vms":"2",
      "max_vms":"10",
      "cooldown":"240",
      "elasticity_policies":[
        {
          "type":"CHANGE",
          "adjust":"5",
          "expression":"ATT=3",
          "period":"5",
          "period_number":"60",
          "cooldown":"240"
        }
      ],
      "scheduled_policies":[
      ]
    }
  ],
  "shutdown_action":"shutdown"
}'

> POST /service_template HTTP/1.1
> Authorization: Basic b25lYWRtaW46b23lbm5lYnVsYQ==
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.14.0.0 zlib/1.2.3 libidn/1.1
> Host: oneflow.server:2474
> Accept: */*
> Content-Length: 771
> Content-Type: application/x-www-form-urlencoded
>
< HTTP/1.1 201 Created
< Content-Type: text/html;charset=utf-8
< X-XSS-Protection: 1; mode=block
< Content-Length: 1990
< X-Frame-Options: sameorigin
< Connection: keep-alive
< Server: thin 1.2.8 codename Black Keys
```

```
<
{
  "DOCUMENT": {
    "TEMPLATE": {
      "BODY": {
        "deployment": "straight",
        "name": "web-application",
        "roles": [
          {
            "scheduled_policies": [
              {
                "adjust": 4,
                "type": "CHANGE",
                "recurrence": "0 2 1-10 * *"
              }
            ],
            "vm_template": 0,
            "name": "frontend",
            "min_vms": 1,
            "max_vms": 4,
            "cardinality": 1,
            "cooldown": 30,
            "shutdown_action": "shutdown",
            "elasticity_policies": [
              {
                "expression": "CUSTOM_ATT>40",
                "adjust": 20,
                "min_adjust_step": 1,
                "cooldown": 30,
                "period": 3,
                "period_number": 30,
                "type": "PERCENTAGE_CHANGE"
              }
            ]
          },
          {
            "scheduled_policies": [

            ],
            "vm_template": 0,
            "name": "worker",
            "min_vms": 2,
            "max_vms": 10,
            "cardinality": 2,
            "parents": [
              "frontend"
            ],
            "cooldown": 240,
            "shutdown_action": "shutdown",
            "elasticity_policies": [
              {
                "expression": "ATT=3",
                "adjust": 5,
                "cooldown": 240,
                "period": 5,
                "period_number": 60,
                "type": "CHANGE"
              }
```

```
            ]
          }
        ],
        "shutdown_action": "shutdown"
      }
    },
    "TYPE": "101",
    "GNAME": "oneadmin",
    "NAME": "web-application",
    "GID": "0",
    "ID": "4",
    "UNAME": "oneadmin",
    "PERMISSIONS": {
      "OWNER_A": "0",
      "OWNER_M": "1",
      "OWNER_U": "1",
      "OTHER_A": "0",
      "OTHER_M": "0",
      "OTHER_U": "0",
      "GROUP_A": "0",
      "GROUP_M": "0",
      "GROUP_U": "0"
    },
    "UID": "0"
  }
```

Get Detailed Information of a Given Service Template

Method	URL	Meaning / Entity Body	Response
GET	/service_template/<id>	**Show** the SERVICE_TEMPLATE resource identified by <id>	**200 OK**: A JSON representation of the collection in the http body

```
curl -u 'oneadmin:opennebula' http://127.0.0.1:2474/service_template/4 -v

> GET /service_template/4 HTTP/1.1
> Authorization: Basic b25lYWRtaW46b3Blbm5lYnVsYQ==
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.14.0.0 zlib/1.2.3 libidn/1.1
> Host: 127.0.0.1:2474
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Type: text/html;charset=utf-8
< X-XSS-Protection: 1; mode=block
< Content-Length: 1990
< X-Frame-Options: sameorigin
< Connection: keep-alive
< Server: thin 1.2.8 codename Black Keys
<
{
  "DOCUMENT": {
    "TEMPLATE": {
      "BODY": {
        "deployment": "straight",
        "name": "web-application",
        "roles": [
          {
            "scheduled_policies": [
```

```
        {
          "adjust": 4,
          "type": "CHANGE",
          "recurrence": "0 2 1-10 * *"
        }
      ],
      "vm_template": 0,
      ...
```

List the Available Service Templates

Method	URL	Meaning / Entity Body	Response
GET	/service_template	**List** the contents of the SERVICE_TEMPLATE collection.	**200 OK**: A JSON representation of the collection in the http body

```
curl -u 'oneadmin:opennebula' http://127.0.0.1:2474/service_template -v

> GET /service_template HTTP/1.1
> Authorization: Basic b25lYWRtaW46b3Blbm5lYnVsYQ==
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.14.0.0 zlib/1.2.3 libidn/1.1
> Host: 127.0.0.1:2474
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Type: text/html;charset=utf-8
< X-XSS-Protection: 1; mode=block
< Content-Length: 6929
< X-Frame-Options: sameorigin
< Connection: keep-alive
< Server: thin 1.2.8 codename Black Keys
<
{
  "DOCUMENT_POOL": {
    "DOCUMENT": [
      {
        "TEMPLATE": {
          "BODY": {
            "deployment": "straight",
            "name": "web-server",
            "roles": [
              {
                "scheduled_policies": [
                  {
                    "adjust": 4,
                    "type": "CHANGE",
                    "recurrence": "0 2 1-10 * *"
                  }
                ],
                "vm_template": 0,
                "name": "frontend",
                "min_vms": 1,
                "max_vms": 4,
                "cardinality": 1,
                "cooldown": 30,
                "shutdown_action": "shutdown",
                "elasticity_policies": [
                  {
```

...

Update a Given Template

Method	URL	Meaning / Entity Body	Response
PUT	/service_template/<id>	**Update** the SERVICE_TEMPLATE resource identified by <id>.	**200 OK**:

```
curl http://127.0.0.1:2474/service_template/4 -u 'oneadmin:opennebula' -v -X PUT --data '{
  "name":"web-application",
  "deployment":"straight",
  "roles":[
    {
      "name":"frontend",
      "cardinality":"1",
      "vm_template":"0",
      "shutdown_action":"shutdown-hard",
      "min_vms":"1",
      "max_vms":"4",
      "cooldown":"30",
      "elasticity_policies":[
        {
          "type":"PERCENTAGE_CHANGE",
          "adjust":"20",
          "min_adjust_step":"1",
          "expression":"CUSTOM_ATT>40",
          "period":"3",
          "period_number":"30",
          "cooldown":"30"
        }
      ],
      "scheduled_policies":[
        {
          "type":"CHANGE",
          "adjust":"4",
          "recurrence":"0 2 1-10 * *"
        }
      ]
    },
    {
      "name":"worker",
      "cardinality":"2",
      "vm_template":"0",
      "shutdown_action":"shutdown",
      "parents":[
        "frontend"
      ],
      "min_vms":"2",
      "max_vms":"10",
      "cooldown":"240",
      "elasticity_policies":[
        {
          "type":"CHANGE",
          "adjust":"5",
          "expression":"ATT=3",
          "period":"5",
```

```
          "period_number":"60",
          "cooldown":"240"
        }
      ],
      "scheduled_policies":[
      ]
    }
  ],
  "shutdown_action":"shutdown"
}'

> PUT /service_template/4 HTTP/1.1
> Authorization: Basic b25lYWRtaW46b3Blbm5lYnVsYQ==
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.14.0.0 zlib/1.2.3 libidn/1.1
> Host: 127.0.0.1:2474
> Accept: */*
> Content-Length: 1219
> Content-Type: application/x-www-form-urlencoded
> Expect: 100-continue
>
* Done waiting for 100-continue
< HTTP/1.1 200 OK
< Content-Type: text/html;charset=utf-8
< X-XSS-Protection: 1; mode=block
< Content-Length: 1995
< X-Frame-Options: sameorigin
< Connection: keep-alive
< Server: thin 1.2.8 codename Black Keys
<
{
  "DOCUMENT": {
    "TEMPLATE": {
      "BODY": {
        "deployment": "straight",
        "name": "web-application",
        "roles": [
          {
            "scheduled_policies": [
              {
                "adjust": 4,
                "type": "CHANGE",
                "recurrence": "0 2 1-10 * *"
              }
            ],
            "vm_template": 0,
            "name": "frontend",
            "min_vms": 1,
            "max_vms": 4,
            "cardinality": 1,
            "cooldown": 30,
            "shutdown_action": "shutdown-hard",
            ...
```

Instantiate a Given Template

Method	URL	Meaning / Entity Body	Response
POST	/service_template/<id>/action	Perform an action on the SERVICE_TEMPLATE resource identified by <id>. Available actions: instantiate, chown, chgrp, chmod	**201**:

Available actions:

- instantiate
- chown
- chmod
- chgrp

```
curl http://127.0.0.1:2474/service_template/4/action -u 'oneadmin:opennebula' -v -X POST --data '{
  "action": {
    "perform":"instantiate"
  }
}'

> POST /service_template/4/action HTTP/1.1
> Authorization: Basic b25lYWRtaW46b3Blbm5lYnVsYQ==
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.14.0.0 zlib/1.2.3 libidn/1.1
> Host: 127.0.0.1:2474
> Accept: */*
> Content-Length: 49
> Content-Type: application/x-www-form-urlencoded
>
< HTTP/1.1 201 Created
< Content-Type: text/html;charset=utf-8
< X-XSS-Protection: 1; mode=block
< Content-Length: 2015
< X-Frame-Options: sameorigin
< Connection: keep-alive
< Server: thin 1.2.8 codename Black Keys
<
{
  "DOCUMENT": {
    "TEMPLATE": {
      "BODY": {
        "deployment": "straight",
        "name": "web-application",
        "roles": [
          {
            "scheduled_policies": [
              {
                "adjust": 4,
                "type": "CHANGE",
                "recurrence": "0 2 1-10 * *"
              }
            ],
            "vm_template": 0,
```

Delete a Given Template

Method	URL	Meaning / Entity Body	Response
DELETE	/service_template/<id>	**Delete** the SERVICE_TEMPLATE resource identified by <id>	**201**:

```
curl http://127.0.0.1:2474/service_template/4 -u 'oneadmin:opennebula' -v -X DELETE

> DELETE /service_template/3 HTTP/1.1
> Authorization: Basic b25lYWRtaW46b3Blbm5lYnVsYQ==
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.14.0.0 zlib/1.2.3 libidn/1.1
> Host: 127.0.0.1:2474
> Accept: */*
>
< HTTP/1.1 201 Created
< Content-Type: text/html;charset=utf-8
< X-XSS-Protection: 1; mode=block
< Content-Length: 0
< X-Frame-Options: sameorigin
< Connection: keep-alive
< Server: thin 1.2.8 codename Black Keys
```

Get Detailed Information of a Given Service

Method	URL	Meaning / Entity Body	Response
GET	/service/<id>	**Show** the SERVICE resource identified by <id>	**200 OK**: A JSON representation of the collection in the http body

```
curl http://127.0.0.1:2474/service/5 -u 'oneadmin:opennebula' -v

> GET /service/5 HTTP/1.1
> Authorization: Basic b25lYWRtaW46b3Blbm5lYnVsYQ==
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.14.0.0 zlib/1.2.3 libidn/1.1
> Host: 127.0.0.1:2474
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Type: text/html;charset=utf-8
< X-XSS-Protection: 1; mode=block
< Content-Length: 11092
< X-Frame-Options: sameorigin
< Connection: keep-alive
< Server: thin 1.2.8 codename Black Keys
<
{
  "DOCUMENT": {
    "TEMPLATE": {
      "BODY": {
        "deployment": "straight",
        "name": "web-application",
        "roles": [
          {
            "scheduled_policies": [
              {
                "adjust": 4,
                "last_eval": 1374676803,
                "type": "CHANGE",
                "recurrence": "0 2 1-10 * *"
```

```
  }
],
"vm_template": 0,
"disposed_nodes": [

],
"name": "frontend",
"min_vms": 1,
"nodes": [
  {
    "deploy_id": 12,
    "vm_info": {
      "VM": {
        "CPU": "33",
        "TEMPLATE": {
          "CPU": "1",
          "CONTEXT": {
            "TARGET": "hda",
            "NETWORK": "YES",
            "DISK_ID": "0"
          },
          "MEMORY": "1024",
          "TEMPLATE_ID": "0",
          "VMID": "12"
        },
        "GNAME": "oneadmin",
        "RESCHED": "0",
        "NET_RX": "1300",
        "NAME": "frontend_0_(service_5)",
        "ETIME": "0",
        "USER_TEMPLATE": {
          "SERVICE_ID": "5",
          "ROLE_NAME": "frontend"
        },
        "GID": "0",
        "LAST_POLL": "1374676793",
        "MEMORY": "786432",
        "HISTORY_RECORDS": {
          "HISTORY": {
            "RETIME": "0",
            "TMMAD": "dummy",
            "DS_LOCATION": "/var/tmp/one_install/var//datastores",
            "SEQ": "0",
            "VNMMAD": "dummy",
            "ETIME": "0",
            "PETIME": "1374676347",
            "HOSTNAME": "vmx_dummy",
            "VMMMAD": "dummy",
            "ESTIME": "0",
            "HID": "2",
            "EETIME": "0",
            "OID": "12",
            "STIME": "1374676347",
            "DS_ID": "0",
            "ACTION": "0",
            "RSTIME": "1374676347",
            "REASON": "0",
            "PSTIME": "1374676347"
```

```
            }
          },
          "ID": "12",
          "DEPLOY_ID": "vmx_dummy:frontend_0_(service_5):dummy",
          "NET_TX": "800",
          "UNAME": "oneadmin",
          "LCM_STATE": "3",
          "STIME": "1374676345",
          "UID": "0",
          "PERMISSIONS": {
            "OWNER_U": "1",
            "OWNER_M": "1",
            "OWNER_A": "0",
            "GROUP_U": "0",
            "GROUP_M": "0",
            "GROUP_A": "0",
            "OTHER_U": "0",
            "OTHER_M": "0",
            "OTHER_A": "0"
          },
          "STATE": "3"
        }
      }
    }
  ],
  "last_vmname": 1,
  "max_vms": 4,
  "cardinality": 1,
  "cooldown": 30,
  "shutdown_action": "shutdown-hard",
  "state": "2",
  "elasticity_policies": [
    {
      "expression": "CUSTOM_ATT>40",
      "true_evals": 0,
      "adjust": 20,
      "min_adjust_step": 1,
      "last_eval": 1374676803,
      "cooldown": 30,
      "expression_evaluated": "CUSTOM_ATT[--] > 40",
      "period": 3,
      "period_number": 30,
      "type": "PERCENTAGE_CHANGE"
    }
  ]
},
{
  "scheduled_policies": [

  ],
  "vm_template": 0,
  "disposed_nodes": [

  ],
  "name": "worker",
  "min_vms": 2,
  "nodes": [
    {
```

```
"deploy_id": 13,
"vm_info": {
  "VM": {
    "CPU": "9",
    "TEMPLATE": {
      "CPU": "1",
      "CONTEXT": {
        "TARGET": "hda",
        "NETWORK": "YES",
        "DISK_ID": "0"
      },
      "MEMORY": "1024",
      "TEMPLATE_ID": "0",
      "VMID": "13"
    },
    "GNAME": "oneadmin",
    "RESCHED": "0",
    "NET_RX": "1600",
    "NAME": "worker_0_(service_5)",
    "ETIME": "0",
    "USER_TEMPLATE": {
      "SERVICE_ID": "5",
      "ROLE_NAME": "worker"
    },
    "GID": "0",
    "LAST_POLL": "1374676783",
    "MEMORY": "545259",
    "HISTORY_RECORDS": {
      "HISTORY": {
        "RETIME": "0",
        "TMMAD": "dummy",
        "DS_LOCATION": "/var/tmp/one_install/var//datastores",
        "SEQ": "0",
        "VNMMAD": "dummy",
        "ETIME": "0",
        "PETIME": "1374676377",
        "HOSTNAME": "xen_dummy",
        "VMMMAD": "dummy",
        "ESTIME": "0",
        "HID": "1",
        "EETIME": "0",
        "OID": "13",
        "STIME": "1374676377",
        "DS_ID": "0",
        "ACTION": "0",
        "RSTIME": "1374676377",
        "REASON": "0",
        "PSTIME": "1374676377"
      }
    },
    "ID": "13",
    "DEPLOY_ID": "xen_dummy:worker_0_(service_5):dummy",
    "NET_TX": "600",
    "UNAME": "oneadmin",
    "LCM_STATE": "3",
    "STIME": "1374676375",
    "UID": "0",
    "PERMISSIONS": {
```

```
          "OWNER_U": "1",
          "OWNER_M": "1",
          "OWNER_A": "0",
          "GROUP_U": "0",
          "GROUP_M": "0",
          "GROUP_A": "0",
          "OTHER_U": "0",
          "OTHER_M": "0",
          "OTHER_A": "0"
        },
        "STATE": "3"
      }
    }
  },
  {
    "deploy_id": 14,
    "vm_info": {
      "VM": {
        "CPU": "75",
        "TEMPLATE": {
          "CPU": "1",
          "CONTEXT": {
            "TARGET": "hda",
            "NETWORK": "YES",
            "DISK_ID": "0"
          },
          "MEMORY": "1024",
          "TEMPLATE_ID": "0",
          "VMID": "14"
        },
        "GNAME": "oneadmin",
        "RESCHED": "0",
        "NET_RX": "1100",
        "NAME": "worker_1_(service_5)",
        "ETIME": "0",
        "USER_TEMPLATE": {
          "SERVICE_ID": "5",
          "ROLE_NAME": "worker"
        },
        "GID": "0",
        "LAST_POLL": "1374676783",
        "MEMORY": "471859",
        "HISTORY_RECORDS": {
          "HISTORY": {
            "RETIME": "0",
            "TMMAD": "dummy",
            "DS_LOCATION": "/var/tmp/one_install/var//datastores",
            "SEQ": "0",
            "VNMMAD": "dummy",
            "ETIME": "0",
            "PETIME": "1374676378",
            "HOSTNAME": "kvm_dummy",
            "VMMMAD": "dummy",
            "ESTIME": "0",
            "HID": "0",
            "EETIME": "0",
            "OID": "14",
            "STIME": "1374676378",
```

```
                    "DS_ID": "0",
                    "ACTION": "0",
                    "RSTIME": "1374676378",
                    "REASON": "0",
                    "PSTIME": "1374676378"
                  }
                },
                "ID": "14",
                "DEPLOY_ID": "kvm_dummy:worker_1_(service_5):dummy",
                "NET_TX": "550",
                "UNAME": "oneadmin",
                "LCM_STATE": "3",
                "STIME": "1374676375",
                "UID": "0",
                "PERMISSIONS": {
                  "OWNER_U": "1",
                  "OWNER_M": "1",
                  "OWNER_A": "0",
                  "GROUP_U": "0",
                  "GROUP_M": "0",
                  "GROUP_A": "0",
                  "OTHER_U": "0",
                  "OTHER_M": "0",
                  "OTHER_A": "0"
                },
                "STATE": "3"
              }
            }
          }
        ],
        "last_vmname": 2,
        "max_vms": 10,
        "cardinality": 2,
        "parents": [
          "frontend"
        ],
        "cooldown": 240,
        "shutdown_action": "shutdown",
        "state": "2",
        "elasticity_policies": [
          {
            "expression": "ATT=3",
            "true_evals": 0,
            "adjust": 5,
            "last_eval": 1374676803,
            "cooldown": 240,
            "expression_evaluated": "ATT[--] = 3",
            "period": 5,
            "period_number": 60,
            "type": "CHANGE"
          }
        ]
      }
    ],
    "log": [
      {
        "message": "New state: DEPLOYING",
        "severity": "I",
```

```
            "timestamp": 1374676345
          },
          {
            "message": "New state: RUNNING",
            "severity": "I",
            "timestamp": 1374676406
          }
        ],
        "shutdown_action": "shutdown",
        "state": 2
      }
    },
    "TYPE": "100",
    "GNAME": "oneadmin",
    "NAME": "web-application",
    "GID": "0",
    "ID": "5",
    "UNAME": "oneadmin",
    "PERMISSIONS": {
      "OWNER_A": "0",
      "OWNER_M": "1",
      "OWNER_U": "1",
      "OTHER_A": "0",
      "OTHER_M": "0",
      "OTHER_U": "0",
      "GROUP_A": "0",
      "GROUP_M": "0",
      "GROUP_U": "0"
    },
    "UID": "0"
  }
```


List the Available Services

Method	URL	Meaning / Entity Body	Response
GET	`/service`	**List** the contents of the `SERVICE` collection.	**200 OK**: A JSON representation of the collection in the http body

```
curl http://127.0.0.1:2474/service -u 'oneadmin:opennebula' -v

> GET /service HTTP/1.1
> Authorization: Basic b25lYWRtaW46b3Blbm5lYnVsYQ==
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.14.0.0 zlib/1.2.3 libidn/1.1
> Host: 127.0.0.1:2474
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Type: text/html;charset=utf-8
< X-XSS-Protection: 1; mode=block
< Content-Length: 12456
< X-Frame-Options: sameorigin
< Connection: keep-alive
< Server: thin 1.2.8 codename Black Keys
<
{
  "DOCUMENT_POOL": {
    "DOCUMENT": [
```

```
{
  "TEMPLATE": {
    "BODY": {
      "deployment": "straight",
      "name": "web-application",
      "roles": [
        {
          "scheduled_policies": [
            {
              "adjust": 4,
              "last_eval": 1374676986,
              "type": "CHANGE",
              "recurrence": "0 2 1-10 * *"
            }
          ],
          ...
```

Perform an Action on a Given Service

Method	URL	Meaning / Entity Body	Response
POST	/service/<id>/action	**Perform** an action on the SERVICE resource identified by <id>.	**201**:

Available actions:

- shutdown: Shutdown a service.
  - From RUNNING or WARNING shuts down the Service
- recover: Recover a failed service, cleaning the failed VMs.
  - From FAILED_DEPLOYING continues deploying the Service
  - From FAILED_SCALING continues scaling the Service
  - From FAILED_UNDEPLOYING continues shutting down the Service
  - From COOLDOWN the Service is set to running ignoring the cooldown duration
  - From WARNING failed VMs are deleted, and new VMs are instantiated
- chown
- chmod
- chgrp

```
curl http://127.0.0.1:2474/service/5/action -u 'oneadmin:opennebula' -v -X POST --data '{
  "action": {
    "perform":"shutdown"
  }
}'

curl http://127.0.0.1:2474/service/5/action -u 'oneadmin:opennebula' -v -X POST --data '{
  "action": {
    "perform":"chgrp",
    "params" : {
      "group_id" : 2
    }
  }
}'
```

Update the Cardinality of a Given Role

Method	URL	Meaning / Entity Body	Response
PUT	/service/<id>/role/<name>	**Update** the ROLE identified by <name> of the SERVICE resource identified by <id>. Currently the only attribute that can be updated is the cardinality.	**200 OK**:

You can force a cardinality outside the defined range with the force param.

```
curl http://127.0.0.1:2474/service/5/role/frontend -u 'oneadmin:opennebula' -X PUT -v --data '{
  "cardinality" : 2,
  "force" : true
}'

> PUT /service/5/role/frontend HTTP/1.1
> Authorization: Basic b25lYWRtaW46b3Blbm5lYnVsYQ==
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.14.0.0 zlib/1.2.3 libidn/1.1
> Host: 127.0.0.1:2474
> Accept: */*
> Content-Length: 41
> Content-Type: application/x-www-form-urlencoded
>
< HTTP/1.1 200 OK
< Content-Type: text/html;charset=utf-8
< X-XSS-Protection: 1; mode=block
< Content-Length: 0
< X-Frame-Options: sameorigin
< Connection: keep-alive
< Server: thin 1.2.8 codename Black Keys
```

Perform an Action on All the VMs of a Given Role

Method	URL	Meaning / Entity Body	Response
POST	/service/<id>/role/<name>/action	**Perform** an action on all the Virtual Machines belonging to the ROLE identified by <name> of the SERVICE resource identified by <id>.	**201**:

You can use this call to perform a VM action on all the Virtual Machines belonging to a role. For example, if you want to suspend the Virtual Machines of the worker Role:

These are the commands that can be performed:

- shutdown
- shutdown-hard
- undeploy
- undeploy-hard
- hold
- release
- stop
- suspend
- resume

- boot
- delete
- delete-recreate
- reboot
- reboot-hard
- poweroff
- poweroff-hard
- snapshot-create

Instead of performing the action immediately on all the VMs, you can perform it on small groups of VMs with these options:

- period: Seconds between each group of actions
- number: Number of VMs to apply the action to each period

```
curl http://127.0.0.1:2474/service/5/role/frontend/action -u 'oneadmin:opennebula' -v -X POST --data
  "action": {
    "perform":"stop",
    "params" : {
      "period" : 60,
      "number" : 2
    }
  }
}'

> POST /service/5/role/frontend/action HTTP/1.1
> Authorization: Basic b25lYWRtaW46b3Blbm5lYnVsYQ==
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.14.0.0 zlib/1.2.3 libidn/1.1
> Host: 127.0.0.1:2474
> Accept: */*
> Content-Length: 106
> Content-Type: application/x-www-form-urlencoded
>
< HTTP/1.1 201 Created
< Content-Type: text/html;charset=utf-8
< X-XSS-Protection: 1; mode=block
< Content-Length: 57
< X-Frame-Options: sameorigin
< Connection: keep-alive
< Server: thin 1.2.8 codename Black Keys
```

CHAPTER

FOUR

INFRASTRUCTURE INTEGRATION

4.1 Using Hooks

The Hook Manager present in OpenNebula enables the triggering of custom scripts tied to a change in state in a particular resource, being that a Host or a Virtual Machine. This opens a wide area of automation for system administrators to tailor their cloud infrastructures.

4.1.1 Configuration

Hook Manager configuration is set in `/etc/one/oned.conf`. Hooks can be tied to changes in host or virtual machine states, and they can be executed locally to the OpenNebula front-end and remotely in the relevant worker node.

In general, hook definition in `/etc/one/oned.conf` has two paremeters:

- **executable**: path of the hook driver executable, can be an absolute path or relative to `/usr/lib/one/mads`
- **arguments**: for the driver executable, can be an absolute path or relative to `/etc/one/`

4.1.2 Hooks for VirtualMachines

In the case of VirtualMachine hooks, the following can be defined:

- **name** : for the hook, useful to track the hook (OPTIONAL)
- **on** : when the hook should be executed,
  - **CREATE**, when the VM is created (onevm create)
  - **RUNNING**, after the VM is successfully booted
  - **SHUTDOWN**, after the VM is shutdown
  - **STOP**, after the VM is stopped (including VM image transfers)
  - **DONE**, after the VM is destroyed or shutdown
  - **UNKNOWN**, when the VM enters the unknown state
  - **FAILED**, when the VM enters the failed state
  - **CUSTOM**, user defined specific STATE and LCM_STATE combination of states to trigger the hook.
- **command** : path can be absolute or relative to `/var/lib/one/remotes/hooks`
- **arguments** : for the hook. You can access the following VM attributes with $

  - **$ID**, the ID of the VM that triggered the hook execution
  - **$TEMPLATE**, the template of the VM that triggered the hook, in xml and base64 encoded
  - **$PREV_STATE**, the previous STATE of the Virtual Machine
  - **$PREV_LCM_STATE**, the previous LCM STATE of the Virtual Machine
- **remote** : values,
  - **YES**, The hook is executed in the host where the VM was allocated
  - **NO**, The hook is executed in the OpenNebula server (default)

The following is an example of a hook tied to the DONE state of a VM:

```
VM_HOOK = [
    name      = "notify_done",
    on        = "DONE",
    command   = "notify.rb",
    arguments = "$ID $TEMPLATE" ]
```

Or an more advanced example:

```
VM_HOOK = [
  name      = "advanced_hook",
  on        = "CUSTOM",
  state     = "ACTIVE",
  lcm_state = "BOOT_UNKNOWN",
  command   = "log.rb",
  arguments = "$ID $PREV_STATE $PREV_LCM_STATE" ]
```

4.1.3 Hooks for Hosts

In the case of Host hooks, the following can be defined:

- **name** : for the hook, useful to track the hook (OPTIONAL)
- **on** : when the hook should be executed,
  - **CREATE**, when the Host is created (onehost create)
  - **ERROR**, when the Host enters the error state
  - **DISABLE**, when the Host is disabled
- **command** : path can be absolute or relative to `/var/lib/one/remotes/hooks`
- **arguments** : for the hook. You can use the following Host attributes with $
  - **$ID**, the ID of the Host that triggered the hook execution
  - **$TEMPLATE**, the full Host information, in xml and base64 encoded
- **remote** : values,
  - **YES**, The hook is executed in the host
  - **NO**, The hook is executed in the OpenNebula server (default)

The following is an example of a hook tied to the ERROR state of a Host:

```
#------------------------------- Host Hook ---------------------------------
# This hook is used to perform recovery actions when a host fails.
# Script to implement host failure tolerance
#   It can be set to
#           -r recreate VMs running in the host
#           -d delete VMs running in the host
#   Additional flags
#           -f force resubmission of suspended VMs
#           -p <n> avoid resubmission if host comes
#                  back after n monitoring cycles
#-------------------------------------------------------------------------------
#
#HOST_HOOK = [
#    name      = "error",
#    on        = "ERROR",
#    command   = "ft/host_error.rb",
#    arguments = "$ID -r",
#    remote    = "no" ]
#-------------------------------------------------------------------------------
```

4.1.4 Other Hooks

Other OpenNebula entities like Virtual Networks, Users, Groups and Images can be hooked on creation and removal. These hooks are specified with the following variables in oned.conf:

- **VNET_HOOK**, for virtual networks
- **USER_HOOK**, for users
- **GROUP_HOOK**, for groups
- **IMAGE_HOOK**, for disk images.

These hooks are always executed on the front-end and are defined by the following attributes

- **name** : for the hook, useful to track the hook (OPTIONAL)
- **on** : when the hook should be executed,
  - **CREATE**, when the object (virtual network, user, group or image) is created
  - **REMOVE**, when the object is removed from the DB
- **command** : path can be absolute or relative to `/var/lib/one/remotes/hooks`
- **arguments** : for the hook. You can use the following Host attributes with $
  - **$ID**, the ID of the Host that triggered the hook execution
  - **$TEMPLATE**, the full Host information, in xml and base64 encoded

The following is an example of a hook that sends and email to a new register user:

```
USER_HOOK = [
    name      = "mail",
    on        = "CREATE",
    command   = "email2user.rb",
    arguments = "$ID $TEMPLATE"]
```

4.1.5 Developing your Hooks

The execution of each hook is tied to the object that trigger the event. The data of the object can be passed to the hook through the $ID and the $TEMPLATE variables:

- $TEMPLATE will give you the full output of the corresponding show command in XML and base64 encoding. This can be easily deal with in any language. If you are using bash for your scripts you may be interested in the xpath.rb util, check the following example:

```
#!/bin/bash
# Argument hook for virtual network add to oned.conf
# VNET_HOOK = [
#   name="bash_arguments",
#   on="CREATE",
#   command=<path_to_this_file>,
#   arguments="$TEMPLATE" ]

XPATH=/var/lib/one/remotes/datastore/xpath.rb
T64=$1

USER_NAME=`$XPATH -b $T64 UNAME`
OWNER_USE_PERMISSION=`$XPATH -b $T64 PERMISSIONS/OWNER_U`

#UNAME and PERMISSIONS/OWNER_U are the XPATH for the attributes without the root element
```

- $ID you can use the ID of the object to retreive more information or to perform an action over the object. (e.g. onevm hold $ID)

Note that within the hook you can further interact with OpenNebula to retrieve more information, or perform any other action

4.2 Virtualization Driver

The component that deals with the hypervisor to create, manage and get information about virtual machine objects is called Virtual Machine Manager (VMM for short). This component has two parts. The first one resides in the core and holds most of the general functionality common to all the drivers (and some specific), the second is the driver that is the one able to translate basic VMM actions to the hypervisor.

4.2.1 Driver Configuration

There are two main drivers `one_vmm_exec` and `one_vmm_sh`. Both take commands from OpenNebula and execute a set of scripts for those actions, the main difference is that `one_vmm_exec` executes the commands remotely (logging into the host where VM is being or is going to be executed) and `one_vmm_sh` does the execution of the scripts in the frontend.

The driver takes some parameters, described here:

parameter	description
-r <num>	number of retries when executing an action
-t <num	number of threads, i.e. number of actions done at the same time
-l <actions>	(`one_vmm_exec` only) actions executed locally, command can be overridden for each action
<driver_directory>	where in the remotes directory the driver can find the action scripts

These are the actions valid in the -l parameter:

- attach_disk

- attach_nic
- cancel
- deploy
- detach_disk
- detach_nic
- kvmrc
- migrate
- migrate_local
- poll
- reboot
- reset
- restore
- save
- shutdown
- snapshot_create
- snapshot_delete
- snapshot_revert

You can also provide an alternative script name for local execution, by default the script is called the same as the action, in this fashion `action=script_name`. As an example:

```
-l migrate,poll=poll_ganglia,save
```

These arguments are specified in the *oned.conf file*, `arguments` variable:

```
VM_MAD = [
    name       = "kvm",
    executable = "one_vmm_exec",
    arguments  = "-t 15 -r 0 -l migrate,save kvm",
    default    = "vmm_exec/vmm_exec_kvm.conf",
    type       = "kvm" ]
```

4.2.2 Actions

Every action should have an executable program (mainly scripts) located in the remote dir (`remotes/vmm/<driver_directory>`) that performs the desired action. These scripts receive some parameters (and in the case of `DEPLOY` also STDIN) and give back the error message or information in some cases writing to STDOUT.

VMM actions, they are the same as the names of the scripts:

- **attach_disk**: Attaches a new DISK in the VM
  - Arguments
    - **DOMAIN**: Domain name: one-101
    - **SOURCE**: Image path
    - **TARGET**: Device in the guest: hda, sdc, vda, xvdc

        * **TARGET_INDEX**: Position in the list of disks
        * **DRV_ACTION**: action xml. Base: `/VMM_DRIVER_ACTION_DATA/VM/TEMPLATE/DISK[ATTACH='YES']`
            - `DRIVER`: Disk format: raw, qcow2
            - `TYPE`: Disk type: block, cdrom, rbd, fs or swap
            - `READONLY`: The value is `YES` when it's read only
            - `CACHE`: Cache mode: none, writethrough, writeback
            - `SOURCE`: Image source, used for ceph
    - Response
        * Success: -
        * Faulure: Error message
- **attach_nic**: Attaches a new NIC in the VM
    - Arguments
        * **DOMAIN**: Domain name: one-808
        * **MAC**: MAC address of the new NIC
        * **BRIDGE**: Bridge where to attach the new NIC
        * **MODEL**: NIC model to emulate, ex: `e1000`
        * **NET_DRV**: Network driver used, ex: `ovswitch`
    - Response
        * Success: -
        * Failure: Error message
- **cancel**: Destroy a VM
    - Arguments:
        * **DOMAIN**: Domain name: one-909
    - Response
        * Success: -
        * Failure: Error message
- **deploy**: Deploy a new VM
    - Arguments:
        * **DEPLOYMENT_FILE**: where to write de deployment file. You have to write whatever comes from STDIN to a file named like this parameter. In shell script you can do: `cat > $domain`
    - Response
        * Success: Deploy id, ex: one-303
        * Failure: Error message
- **detach_disk**: Detaches a DISK from a VM
    - Arguments
        * **DOMAIN**: Domain name: one-286

        - **SOURCE**: Image path
        - **TARGET**: Device in the guest: hda, sdc, vda, xvdc
        - **TARGET_INDEX**: Position in the list of disks
    - Response
        - Success: -
        - Failure: Error message
- **detach_nic**: Detaches a NIC from a VM
    - Arguments
        - **DOMAIN**: Domain name: one-286
        - **MAC**: MAC address of the NIC to detach
    - Response
        - Success: -
        - Failure: Error message
- **migrate**: Live migrate a VM to another host
    - Arguments:
        - **DOMAIN**: Domain name: one-286
        - **DESTINATION_HOST**: Host where to migrate the VM
        - **HOST**: Host where the VM is currently running
    - Response
        - Success: -
        - Failure: Error message
- **poll**: Get information from a VM
    - Arguments:
        - **DOMAIN**: Domain name: one-286
        - **HOST**: Host where the VM is running
    - Response
        - Success: -
        - Failure: Error message
- **reboot**: Orderly reboots a VM
    - Arguments:
        - **DOMAIN**: Domain name: one-286
        - **HOST**: Host where the VM is running
    - Response
        - Success: -
        - Failure: Error message
- **reset**: Hard reboots a VM

  - Arguments:
    - **DOMAIN**: Domain name: one-286
    - **HOST**: Host where the VM is running
  - Response
    - Success: -
    - Failure: Error message
- **restore**: Restores a previously saved VM
  - Arguments:
    - **FILE**: VM save file
    - **HOST**: Host where to restore the VM
  - Response
    - Success: -
    - Failure: Error message
- **save**: Saves a VM
  - Arguments:
    - **DOMAIN**: Domain name: one-286
    - **FILE**: VM save file
    - **HOST**: Host where the VM is running
  - Response
    - Success: -
    - Failure: Error message
- **shutdown**: Orderly shutdowns a VM
  - Arguments:
    - **DOMAIN**: Domain name: one-286
    - **HOST**: Host where the VM is running
  - Response
    - Success: -
    - Failure: Error message
- **snapshot_create**: Makes a new snapshot of a VM
  - Arguments:
    - **DOMAIN**: Domain name: one-286
    - **ONE_SNAPSHOT_ID**: OpenNebula snapshot identifier
  - Response
    - Success: Snapshot name for the hypervisor. Used later to delete or revert
    - Failure: Error message
- **snapshot_delete**: Deletes a snapshot of a VM

  - Arguments:
    - **DOMAIN**: Domain name: one-286
    - **SNAPSHOT_NAME**: Name used to refer the snapshot in the hypervisor
  - Response
    - Success: -
    - Failure: Error message
- **snapshot_revert**: Returns a VM to an saved state
  - Arguments:
    - **DOMAIN**: Domain name: one-286
    - **SNAPSHOT_NAME**: Name used to refer the snapshot in the hypervisor
  - Response
    - Success: -
    - Failure: Error message

`action xml` parameter is a base64 encoded xml that holds information about the VM. To get one of the values explained in the documentation, for example from `attach_disk READONLY` you can add to the base XPATH the name of the parameter. XPATH:

```
/VMM_DRIVER_ACTION_DATA/VM/TEMPLATE/DISK[ATTACH='YES']/READONLY
```

When using shell script there is a handy script that gets parameters for given XPATH in that XML. Example:

```
XPATH="${DRIVER_PATH}/../../datastore/xpath.rb -b $DRV_ACTION"

unset i j XPATH_ELEMENTS

DISK_XPATH="/VMM_DRIVER_ACTION_DATA/VM/TEMPLATE/DISK[ATTACH='YES']"

while IFS= read -r -d '' element; do
    XPATH_ELEMENTS[i++]="$element"
done < <($XPATH     $DISK_XPATH/DRIVER \
                    $DISK_XPATH/TYPE \
                    $DISK_XPATH/READONLY \
                    $DISK_XPATH/CACHE \
                    $DISK_XPATH/SOURCE)

DRIVER="${XPATH_ELEMENTS[j++]:-$DEFAULT_TYPE}"
TYPE="${XPATH_ELEMENTS[j++]}"
READONLY="${XPATH_ELEMENTS[j++]}"
CACHE="${XPATH_ELEMENTS[j++]}"
IMG_SRC="${XPATH_ELEMENTS[j++]}"
```

`one_vmm_sh` has the same script actions and meanings but an argument more that is the host where the action is going to be performed. This argument is always the first one. If you use `-p` parameter in `one_vmm_ssh` the poll action script is called with one more argument that is the host where the VM resides, also it is the same parameter.

4.2.3 Poll Information

`POLL` is the action that gets monitoring info from the running VMs. The format it is supposed to give back information is a line with `KEY=VALUE` pairs separated by spaces. Like this:

```
STATE=a USEDMEMORY=554632
```

The poll action can give back any information and it will be added to the VM information hold but there are some variables that should be given back as they are meaningful to OpenNebula:

Variable	Description
STATE	State of the VM (explained later)
USEDCPU	Percentage of 1 CPU consumed (two fully consumed cpu is 200)
USEDMEMORY	Memory consumption in kilobytes
NETRX	Received bytes from the network
NETTX	Sent bytes to the network

`STATE` is a single character that tells OpenNebula the status of the VM, the states are the ones in this table:

state	description
N/A	Detecting state error. The monitoring could be done, but it returned an unexpected output.
a	Active. The VM is alive, but not necessary running. Could be blocked, booting, etc.
p	Paused. Self-explanatory.
e	Error. The VM crashed or somehow its deployment failed.
d	Disappeared. The VM is not known by the hypervisor anymore.

4.2.4 Deployment File

The deployment file is a text file written by OpenNebula core that holds the information of a VM. It is used when deploying a new VM. OpenNebula is able to generate three formats of deployment files:

- **xen**: deployment file suitable to be used with xen tools
- **kvm**: libvirt format used to create kvm VMs
- **xml**: xml representation of the VM

If the target hypervisor is not xen nor libvirt/kvm the best format to use is xml as it holds more information than the two others. It has all the template information encoded as xml. This is an example:

```
<TEMPLATE>
  <CPU><![CDATA[1.0]]></CPU>
  <DISK>
    <DISK_ID><![CDATA[0]]></DISK_ID>
    <SOURCE><![CDATA[/home/user/vm.img]]></SOURCE>
    <TARGET><![CDATA[sda]]></TARGET>
  </DISK>
  <MEMORY><![CDATA[512]]></MEMORY>
  <NAME><![CDATA[test]]></NAME>
  <VMID><![CDATA[0]]></VMID>
</TEMPLATE>
```

There are some information added by OpenNebula itself like the VMID and the `DISK_ID` for each disk. `DISK_ID` is very important as the disk images are previously manipulated by the `TM` driver and the disk should be accessible at `VM_DIR/VMID/images/disk.DISK_ID`.

4.3 Storage Driver

The Storage subsystem is highly modular. These drivers are separated into two logical sets:

- **DS**: Datastore drivers. They serve the purpose of managing images: register, delete, and create empty datablocks.

- **TM**: Transfer Manager drivers. They manage images associated to instantiated VMs.

4.3.1 Datastore Drivers Structure

Located under `/var/lib/one/remotes/datastore/<ds_mad>`

- **cp**: copies/dumps the image to the datastore
  - **ARGUMENTS**: `datastore_image_dump image_id`
  - **RETURNS**: `image_source size`
  - `datastore_image_dump` is an XML dump of the driver action encoded in Base 64. See a decoded *example*.
  - `image_source` is the image source which will be later sent to the transfer manager
- **mkfs**: creates a new empty image in the datastore
  - **ARGUMENTS**: `datastore_image_dump image_id`
  - **RETURNS**: `image_source size`
  - `datastore_image_dump` is an XML dump of the driver action encoded in Base 64. See a decoded *example*.
  - `image_source` is the image source which will be later sent to the transfer manager.
- **rm**: removes an image fromthe datastore
  - **ARGUMENTS**: `datastore_image_dump image_id`
  - **RETURNS**: –
  - `datastore_image_dump` is an XML dump of the driver action encoded in Base 64. See a decoded *example*.
- **stat**: returns the size of an image in Mb
  - **ARGUMENTS**: `datastore_image_dump image_id`
  - **RETURNS**: `size`
  - `datastore_image_dump` is an XML dump of the driver action encoded in Base 64. See a decoded *example*.
  - `size` the size of the image in Mb.
- **clone**: clones an image.
  - **ARGUMENTS**: `datastore_action_dump image_id`
  - **RETURNS**: `source`
  - `datastore_image_dump` is an XML dump of the driver action encoded in Base 64. See a decoded *example*.
  - `source` the new `source` for the image.
- **monitor**: monitors a datastore
  - **ARGUMENTS**: `datastore_action_dump image_id`
  - **RETURNS**: `monitor data`
  - `datastore_image_dump` is an XML dump of the driver action encoded in Base 64. See a decoded *example*.

  - `monitor data` The monitoring information of the datastore, namely "USED_MB=...\nTOTAL_MB=...\nFREE_MB=..." which are respectively the used size of the datastore in MB, the total capacity of the datastore in MB and the available space in the datastore in MB.

Warning: `image_source` has to be dynamically generated by the `cp` and `mkfs` script. It will be passed later on to the transfer manager, so it should provide all the information the transfer manager needs to locate the image. For instance, in FS_DRIVERS: `DATASTORE_BASE_PATH + md5sum(date + id).`

4.3.2 TM Drivers Structure

This is a list of the TM drivers and their action. Note that they don't return anything. If the exit code is not `0`, the driver will have failed.

Located under `/var/lib/one/remotes/tm/<tm_mad>`. There are two types of action scripts: the first group applies to general image datastores and includes (`clone`, `ln`, `mv` and `mvds`); the second one is only used in conjunction with the system datastore.

Action scripts for generic image datastores:

- **clone**: clones the image from the datastore (non-persistent images)
  - **ARGUMENTS**: `fe:SOURCE host:remote_system_ds/disk.i vm_id ds_id`
  - `fe` is the front-end hostname
  - `SOURCE` is the path of the disk image in the form DS_BASE_PATH/disk
  - `host` is the target host to deploy the VM
  - `remote_system_ds` is the path for the system datastore in the host
  - `vm_id` is the id of the VM
  - `ds_id` is the target datastore (the system datastore)
- **ln**: Links the image from the datastore (persistent images)
  - **ARGUMENTS**: `fe:SOURCE host:remote_system_ds/disk.i vm_id ds_id`
  - `fe` is the front-end hostname
  - `SOURCE` is the path of the disk image in the form DS_BASE_PATH/disk
  - `host` is the target host to deploy the VM
  - `remote_system_ds` is the path for the system datastore in the host
  - `vm_id` is the id of the VM
  - `ds_id` is the target datastore (the system datastore)
- **mvds**: moves an image back to its datastore (persitent images or deferred snapshots)
  - **ARGUMENTS**: `host:remote_system_ds/disk.i fe:SOURCE vm_id ds_id`
  - `fe` is the front-end hostname
  - `SOURCE` is the path of the disk image in the form DS_BASE_PATH/disk
  - `host` is the target host to deploy the VM
  - `remote_system_ds` is the path for the system datastore in the host
  - `vm_id` is the id of the VM

  - `ds_id` is the target datastore (the original datastore for the image)
- **cpds**: moves an image back to its datastore (executed for life disk snapshots)
  - **ARGUMENTS**: `host:remote_system_ds/disk.i fe:SOURCE vm_id ds_id`
  - `fe` is the front-end hostname
  - `SOURCE` is the path of the disk image in the form DS_BASE_PATH/disk
  - `host` is the target host to deploy the VM
  - `remote_system_ds` is the path for the system datastore in the host
  - `vm_id` is the id of the VM
  - `ds_id` is the target datastore (the original datastore for the image)

Action scripts needed when the TM is used for the system datastore:

- **context**: creates an ISO that contains all the files passed as an argument.
  - **ARGUMENTS**: `file1 file2 ... fileN host:remote_system_ds/disk.i vm_id ds_id`
  - `host` is the target host to deploy the VM
  - `remote_system_ds` is the path for the system datastore in the host
  - `vm_id` is the id of the VM
  - `ds_id` is the target datastore (the system datastore)
- **delete**: removes the eithr system datastore's directory of the VM or a disk itself.
  - **ARGUMENTS**: `host:remote_system_ds/disk.i|host:remote_system_ds/ vm_id ds_id`
  - `host` is the target host to deploy the VM
  - `remote_system_ds` is the path for the system datastore in the host
  - `vm_id` is the id of the VM
  - `ds_id` is the target datastore (the system datastore)
- **mkimage**: creates an image on-the-fly bypassing the datastore/image workflow
  - **ARGUMENTS**: `size format host:remote_system_ds/disk.i vm_id ds_id`
  - `size` size in MB of the image
  - `format` format for the image
  - `host` is the target host to deploy the VM
  - `remote_system_ds` is the path for the system datastore in the host
  - `vm_id` is the id of the VM
  - `ds_id` is the target datastore (the system datastore)
- **mkswap**: creates a swap image
  - **ARGUMENTS**: `size host:remote_system_ds/disk.i vm_id ds_id`
  - `size` size in MB of the image
  - `host` is the target host to deploy the VM
  - `remote_system_ds` is the path for the system datastore in the host

  - `vm_id` is the id of the VM
  - `ds_id` is the target datastore (the system datastore)
- **mv**: moves images/directories across system_ds in different hosts. When used for the system datastore the script will received the directory ARGUMENT
  - **ARGUMENTS**: `hostA:system_ds/disk.i|hostB:system_ds/disk.i vm_id ds_id` OR `hostA:system_ds/|hostB:system_ds/ vm_id ds_id`
  - `hostA` is the host the VM is in.
  - `hostB` is the target host to deploy the VM
  - `system_ds` is the path for the system datastore in the host
  - `vm_id` is the id of the VM
  - `ds_id` is the target datastore (the system datastore)
- **premigrate**: It is executed before a livemigration operation is issued to the hypervisor. Note that **only the premigrate script from the system datastore will be used**. Any customization must be done for the premigrate script of the system datastore, although you will probably add operations for other backends than that used by the system datastore.
  - **ARGUMENTS**: `source_host dst_host remote_system_dir vmid dsid template`
  - `src_host` is the host the VM is in.
  - `dst_host` is the target host to migrate the VM to
  - `remote_system_ds_dir` is the path for the VM directory in the system datastore in the host
  - `vmid` is the id of the VM
  - `dsid` is the target datastore
  - `template` is the template of the VM in XML and base64 encoded
- **postmigrate**: It is executed after a livemigration operation. Note that **only the postmigrate script from the system datastore will be used**. Any customization must be done for the postmigrate script of the system datastore, although you will probably add operations for other backends than that used by the system datastore.
  - **ARGUMENTS**: `source_host dst_host remote_system_dir vmid dsid template`
  - see `premigrate` description.

Warning: You only need to implement one `mv` script, but consider the arguments received when the TM is used for the system datastore, a regular image datastore or both.

Warning: If the TM is only for regular images you only need to implement the first group.

4.3.3 An Example VM

Consider a VM with two disks:

```
NAME   = vm01
CPU    = 0.1
MEMORY = 64

DISK   = [ IMAGE_ID = 0 ] # non-persistent disk
DISK   = [ IMAGE_ID = 1 ] # persistent disk
```

This a list of TM actions that will be called upon the events listed:

CREATE

```
<tm_mad>/clone <frontend>:<non_pers_image_source> <host01>:<ds_path>/<vm_id>/disk.0
<tm_mad>/ln <frontend>:<pers_image_source> <host01>:<ds_path>/<vm_id>/disk.1
```

STOP

```
<tm_mad>/mv <host01>:<ds_path>/<vm_id>/disk.0 <frontend>:<ds_path>/<vm_id>/disk.0
<tm_mad>/mv <host01>:<ds_path>/<vm_id>/disk.1 <frontend>:<ds_path>/<vm_id>/disk.1
<tm_mad_sysds>/mv <host01>:<ds_path>/<vm_id> <frontend>:<ds_path>/<vm_id>
```

RESUME

```
<tm_mad>/mv <frontend>:<ds_path>/<vm_id>/disk.0 <host01>:<ds_path>/<vm_id>/disk.0
<tm_mad>/mv <frontend>:<ds_path>/<vm_id>/disk.1 <host01>:<ds_path>/<vm_id>/disk.1
<tm_mad_sysds>/mv <frontend>:<ds_path>/<vm_id> <host01>:<ds_path>/<vm_id>
```

MIGRATE host01 → host02

```
<tm_mad>/mv <host01>:<ds_path>/<vm_id>/disk.0 <host02>:<ds_path>/<vm_id>/disk.0
<tm_mad>/mv <host01>:<ds_path>/<vm_id>/disk.1 <host02>:<ds_path>/<vm_id>/disk.1
<tm_mad_sysds>/mv <host01>:<ds_path>/<vm_id> <host02>:<ds_path>/<vm_id>
```

SHUTDOWN

```
<tm_mad>/delete <host02>:<ds_path>/<vm_id>/disk.0
<tm_mad>/mvds <host02>:<ds_path>/<vm_id>/disk.1 <pers_image_source>
<tm_mad_sysds>/delete <host02>:<ds_path>/<vm_id>
```

- `non_pers_image_source`: Source of the non persistent image.
- `pers_image_source` : Source of the persistent image.
- `frontend`: hostname of the frontend
- `host01`: hostname of host01
- `host02`: hostname of host02
- `tm_mad`: TM driver of the datastore where the image is registered
- `tm_mad_sysds`: TM driver of the system datastore

4.3.4 Helper Scripts

There is a helper shell script with some functions defined to do some common tasks. It is located in `/var/lib/one/remotes/scripts_common.sh`

Here are the description of those functions.

- **log**: Takes one parameter that is a message that will be logged into the VM log file.

```
log "Creating directory $DST_DIR"
```

- **error_message**: sends an exit message to oned surrounding it by separators, use to send the error message when a command fails.

```
error_message "File '$FILE' not found"
```

- **arg_host**: gets the hostname part from a parameter

```
SRC_HOST=`arg_host $SRC`
```

- **arg_path**: gets the path part from a parameter

```
SRC_PATH=`arg_path $SRC`
```

- **exec_and_log**: executes a command and logs its execution. If the command fails the error message is sent to oned and the script ends

```
exec_and_log "chmod g+w $DST_PATH"
```

- **ssh_exec_and_log**: This function executes $2 at $1 host and report error $3

```
ssh_exec_and_log "$HOST" "chmod g+w $DST_PATH" "Error message"
```

- **timeout_exec_and_log**: like `exec_and_log` but takes as first parameter the max number of seconds the command can run

```
timeout_exec_and_log 15 "cp $SRC_PATH $DST_PATH"
```

The are additional minor helper functions, please read the `scripts_common.sh` to see them.

4.3.5 Decoded Example

```
<DS_DRIVER_ACTION_DATA>
    <IMAGE>
        <ID>0</ID>
        <UID>0</UID>
        <GID>0</GID>
        <UNAME>oneadmin</UNAME>
        <GNAME>oneadmin</GNAME>
        <NAME>ttylinux</NAME>
        <PERMISSIONS>
            <OWNER_U>1</OWNER_U>
            <OWNER_M>1</OWNER_M>
            <OWNER_A>0</OWNER_A>
            <GROUP_U>0</GROUP_U>
            <GROUP_M>0</GROUP_M>
            <GROUP_A>0</GROUP_A>
            <OTHER_U>0</OTHER_U>
            <OTHER_M>0</OTHER_M>
            <OTHER_A>0</OTHER_A>
        </PERMISSIONS>
        <TYPE>0</TYPE>
        <DISK_TYPE>0</DISK_TYPE>
        <PERSISTENT>0</PERSISTENT>
        <REGTIME>1385145541</REGTIME>
        <SOURCE/>
        <PATH>/tmp/ttylinux.img</PATH>
        <FSTYPE/>
        <SIZE>40</SIZE>
        <STATE>4</STATE>
        <RUNNING_VMS>0</RUNNING_VMS>
        <CLONING_OPS>0</CLONING_OPS>
        <CLONING_ID>-1</CLONING_ID>
        <DATASTORE_ID>1</DATASTORE_ID>
        <DATASTORE>default</DATASTORE>
        <VMS/>
```

```
        <CLONES/>
        <TEMPLATE>
            <DEV_PREFIX><![CDATA[hd]]></DEV_PREFIX>
            <PUBLIC><![CDATA[YES]]></PUBLIC>
        </TEMPLATE>
    </IMAGE>
    <DATASTORE>
        <ID>1</ID>
        <UID>0</UID>
        <GID>0</GID>
        <UNAME>oneadmin</UNAME>
        <GNAME>oneadmin</GNAME>
        <NAME>default</NAME>
        <PERMISSIONS>
            <OWNER_U>1</OWNER_U>
            <OWNER_M>1</OWNER_M>
            <OWNER_A>0</OWNER_A>
            <GROUP_U>1</GROUP_U>
            <GROUP_M>0</GROUP_M>
            <GROUP_A>0</GROUP_A>
            <OTHER_U>1</OTHER_U>
            <OTHER_M>0</OTHER_M>
            <OTHER_A>0</OTHER_A>
        </PERMISSIONS>
        <DS_MAD>fs</DS_MAD>
        <TM_MAD>shared</TM_MAD>
        <BASE_PATH>/var/lib/one//datastores/1</BASE_PATH>
        <TYPE>0</TYPE>
        <DISK_TYPE>0</DISK_TYPE>
        <CLUSTER_ID>-1</CLUSTER_ID>
        <CLUSTER/>
        <TOTAL_MB>86845</TOTAL_MB>
        <FREE_MB>20777</FREE_MB>
        <USED_MB>1000</USED_MB>
        <IMAGES/>
        <TEMPLATE>
            <CLONE_TARGET><![CDATA[SYSTEM]]></CLONE_TARGET>
            <DISK_TYPE><![CDATA[FILE]]></DISK_TYPE>
            <DS_MAD><![CDATA[fs]]></DS_MAD>
            <LN_TARGET><![CDATA[NONE]]></LN_TARGET>
            <TM_MAD><![CDATA[shared]]></TM_MAD>
            <TYPE><![CDATA[IMAGE_DS]]></TYPE>
        </TEMPLATE>
    </DATASTORE>
</DS_DRIVER_ACTION_DATA>
```

4.4 Monitoring Driver

The Monitoring Drivers (or IM drivers) collect host and virtual machine monitoring data by executing a set of probes in the hosts. This data is either actively queried by OpenNebula or sent periodically by an agent running in the hosts to the frontend.

This guide describes the process of customize or add probes to the hosts. It is also a starting point on how to create a new IM driver from scratch.

4.4.1 Probe Location

The default probes are installed in the frontend in the following path:

- **KVM and Xen**: `/var/lib/one/remotes/im/<hypervisor>-probes.d`
- **VMware and EC2**: `/var/lib/one/remotes/im/<hypervisor>.d`

In the case of `KVM` and `Xen`, the probes are distributed to the hosts, therefore if the probes are changed, they **must** be distributed to the hosts by running `onehost sync`.

4.4.2 General Probe Structure

An IM diver is composed of one or several scripts that write to `stdout` information in this form:

```
KEY1="value"
KEY2="another value with spaces"
```

The drivers receive the following parameters:

Position	Description
1	**hypervisor**: The name of the hypervisor: `kvm`, `xen`, etc...
2	**datastore location**: path of the datastores directory in the host
3	**collectd port**: port in which the `collectd` is listening on
4	**monitor push cycle**: time in seconds between monitorization actions for the UDP-push model
5	**host_id**: id of the host
6	**host_name**: name of the host

Take into account that in shell script the parameters start at 1 (`$1`) and in ruby start at 0 (`ARGV[0]`). For shell script you can use this snippet to get the parameters:

```
hypervisor=$1
datastore_location=$2
collectd_port=$3
monitor_push_cycle=$4
host_id=$5
host_name=$6
```

4.4.3 Basic Monitoring Scripts

You can add any key and value you want to use later in `RANK` and `REQUIREMENTS` for scheduling but there are some basic values you should output:

Key	Description
HYPERVISOR	Name of the hypervisor of the host, useful for selecting the hosts with an specific technology.
TOTALCPU	Number of CPUs multiplied by 100. For example, a 16 cores machine will have a value of 1600.
CPUSPEED	Speed in Mhz of the CPUs.
TOTALMEMORY	Maximum memory that could be used for VMs. It is advised to take out the memory used by the hypervisor.
USEDMEMORY	Memory used, in kilobytes.
FREEMEMORY	Available memory for VMs at that moment, in kilobytes.
FREECPU	Percentage of idling CPU multiplied by the number of cores. For example, if 50% of the CPU is idling in a 4 core machine the value will be 200.
USEDCPU	Percentage of used CPU multiplied by the number of cores.
NETRX	Received bytes from the network
NETTX	Transferred bytes to the network

For example, a probe that gets memory information about a host could be something like:

```
#!/bin/bash

total=$(free | awk ' /^Mem/ { print $2 }')
used=$(free | awk '/buffers\/cache/ { print $3 }')
free=$(free | awk '/buffers\/cache/ { print $4 }')

echo "TOTALMEMORY=$total"
echo "USEDMEMORY=$used"
echo "FREEMEMORY=$free"
```

Executing it should give use memory values:

```
$ ./memory_probe
TOTALMEMORY=1020696
USEDMEMORY=209932
FREEMEMORY=810724
```

For real examples check the directories at `/var/lib/one/remotes/im`.

4.4.4 VM Information

The scripts should also provide information about the VMs running in the host. This is useful as it will only need one call to gather all that information about the VMs in each host. The output should be in this form:

```
VM_POLL=YES
VM=[
  ID=86,
  DEPLOY_ID=one-86,
  POLL="USEDMEMORY=918723 USEDCPU=23 NETTX=19283 NETRX=914 STATE=a" ]
VM=[
  ID=645,
  DEPLOY_ID=one-645,
  POLL="USEDMEMORY=563865 USEDCPU=74 NETTX=2039847 NETRX=2349923 STATE=a" ]
```

The first line (`VM_POLL=YES`) is used to indicate OpenNebula that VM information will follow. Then the information about the VMs is output in that form.

Key	Description
ID	OpenNebula VM id. It can be -1 in case this VM was not created by OpenNebula
DEPLOY_ID	Hypervisor name or identifier of the VM
POLL	VM monitoring info, in the same format as *VMM driver* poll

For example here is a simple script to get qemu/kvm VMs status from libvirt. As before, check the scripts from OpenNebula for a complete example:

```
#!/bin/bash

echo "VM_POLL=YES"

virsh -c qemu:///system list | grep one- | while read vm; do
    deploy_id=$(echo $vm | cut -d' ' -f 2)
    id=$(echo $deploy_id | cut -d- -f 2)
    status_str=$(echo $vm | cut -d' ' -f 3)

    if [ $status_str == "running" ]; then
        state="a"
    else
        state="e"
    fi

    echo "VM=["
    echo "  ID=$id,"
    echo "  DEPLOY_ID=$deploy_id,"
    echo "  POLL=\"STATE=$state\" ]"
done
```

```
$ ./vm_poll
VM_POLL=YES
VM=[
  ID=0,
  DEPLOY_ID=one-0,
  POLL="STATE=a" ]
VM=[
  ID=1,
  DEPLOY_ID=one-1,
  POLL="STATE=a" ]
```

4.4.5 System Datastore Information

Information Manager drivers are also responsible to collect the datastore sizes and its available space. To do so there is an already made script that collects this information for filesystem and lvm based datastores. You can copy it from the KVM driver (`/var/lib/one/remotes/im/kvm-probes.d/monitor_ds.sh`) into your driver directory.

In case you want to create your own datastore monitor you have to return something like this in STDOUT:

```
DS_LOCATION_USED_MB=1
DS_LOCATION_TOTAL_MB=12639
DS_LOCATION_FREE_MB=10459
DS = [
  ID = 0,
  USED_MB = 1,
  TOTAL_MB = 12639,
  FREE_MB = 10459
]
DS = [
```

```
  ID = 1,
  USED_MB = 1,
  TOTAL_MB = 12639,
  FREE_MB = 10459
]
DS = [
  ID = 2,
  USED_MB = 1,
  TOTAL_MB = 12639,
  FREE_MB = 10459
]
```

These are the meanings of the values:

Variable	Description
DS_LOCATION_USED_MB	Used space in megabytes in the DATASTORE LOCATION
DS_LOCATION_TOTAL_MB	Total space in megabytes in the DATASTORE LOCATION
DS_LOCATION_FREE_MB	FREE space in megabytes in the DATASTORE LOCATION
ID	ID of the datastore, this is the same as the name of the directory
USED_MB	Used space in megabytes for that datastore
TOTAL_MB	Total space in megabytes for that datastore
FREE_MB	Free space in megabytes for that datastore

The DATASTORE LOCATION is the path where the datastores are mounted. By default is `/var/lib/one/datastores` but it is specified in the second parameter of the script call.

4.4.6 Creating a New IM Driver

Choosing the Execution Engine

OpenNebula provides two IM probe execution engines: `one_im_sh` and `one_im_ssh`. `one_im_sh` is used to execute probes in the frontend, for example `vmware` uses this engine as it collects data via an API call executed in the frontend. On the other hand, `one_im_ssh` is used when probes need to be run remotely in the hosts, which is the case for `Xen` and `KVM`.

Populating the Probes

Both `one_im_sh` and `one_im_ssh` require an argument which indicates the directory that contains the probes. This argument is appended with ".d".

Example: For `VMware` the execution engine is `one_im_sh` (local execution) and the argument is `vmware`, therefore the probes that will be executed in the hosts are located in `/var/lib/one/remotes/im/vmware.d`

Making Use of Colllectd

If the new IM driver wishes to use the `collectd` component, it needs to:

- Use `one_im_ssh`
- The `/var/lib/one/remotes/im/<im_name>.d` should **only** contain 2 files, the sames that are provided by default inside `kvm.d` and `xen.d`, which are: `collectd-client_control.sh` and `collectd-client.rb`.
- The probes should be actually placed in the `/var/lib/one/remotes/im/<im_name>-probes.d` folder.

Enabling the Driver

A new IM section should be placed added to `oned.conf`.

Example:

```
IM_MAD = [
      name       = "ganglia",
      executable = "one_im_sh",
      arguments  = "ganglia" ]
```

4.5 Networking Driver

This component is in charge of configuring the network in the hypervisors. The purpose of this guide is to describe how to create a new network manager driver.

4.5.1 Driver Configuration and Description

To enable a new network manager driver, the only requirement is to make a new directory with the name of the driver in `/var/lib/one/remotes/vnm/remotes/<name>` with three files:

- **Pre**: This driver should perform all the network related actions required before the Virtual Machine starts in a host.
- **Post**: This driver should perform all the network related actions required after the Virtual Machine starts (actions which typically require the knowledge of the `tap` interface the Virtual Machine is connected to).
- **Clean**: If any clean-up should be performed after the Virtual Machine shuts down, it should be placed here.

Warning: The above three files **must exist**. If no action is required in them a simple `exit 0` will be enough.

Virtual Machine actions and their relation with Network actions:

- **Deploy**: `pre` and `post`
- **Shutdown**: `clean`
- **Cancel**: `clean`
- **Save**: `clean`
- **Restore**: `pre` and `post`
- **Migrate**: `pre` (target host), `clean` (source host), `post` (target host)
- **Attach Nic**: `pre` and `post`
- **Detach Nic**: `clean`

4.5.2 Driver Paramenters

All three driver actions have a first parameter which is the XML VM template encoded in base64 format.

Additionally the `post` driver has a second parameter which is the deploy-id of the Virtual Machine e.g.: `one-17`.

4.6 Authentication Driver

This guide will show you how to develop a new driver for OpenNebula to interact with an external authentication service.

OpenNebula comes with an internal user/password way of authentication, this is called `core`. To be able to use other auth methods there is a system that performs authentication with external systems. Authentication drivers are responsible of getting the user credentials from OpenNebula database and login and answer whether the authentication is correct or not.

In the OpenNebula database there are two values saved for every user, this is `username` and `password`. When the driver used for authentication is core (authenticated without an external auth driver) the password value holds the SHA1 hash of the user's password. In case we are using other authentication method this `password` field can contain any other information we can use to recognize a user, for example, for x509 authentication this field contains the user's public key.

4.6.1 Authentication Driver

Authentication drivers are located at `/var/lib/one/remotes/auth`. There is a directory for each of authentication drivers with an executable inside called `authenticate`. The name of the directory has to be the same as the user's auth driver we want to authenticate. For example, if a user has as auth driver `x509` OpenNebula will execute the file `/var/lib/one/remotes/auth/x509/authenticate` when he performs an OpenNebula action.

The script receives three parameters:

- `username`: name of the user who wants to authenticate.
- `password`: value of the password field for the user that is trying to authenticate. This can be – when the user does not exist in the OpenNebula database.
- `secret`: value provided in the password field of the authentication string.

For example, we can create a new authentication method that just checks the length of the password. For this we can store in the password field the number of characters accepted, for example 5, and user name test. Here are some example calls to the driver with several passwords:

```
authenticate test 5 testpassword
authenticate test 5 another_try
authenticate test 5 12345
```

The script should exit with a non 0 status when the authentication is not correct and write in `stderr` the error. When the authentication is correct it should return:

- Name of the driver. This is used when the user does not exist, this will be written to user's the auth driver field.
- User name
- Text to write in the user's password field in case the user does not exist.

The code for the `/var/lib/one/remotes/auth/length/authenticate` executable can be:

```
#!/bin/bash

username=$1
password=$2
secret=$3

length=$(echo -n "$secret" | wc -c | tr -d ' ')

if [ $length = $password ]; then
```

```
    echo "length $username $secret"
else
    echo "Invalid password"
    exit 255
fi
```

4.6.2 Enabling the Driver

To be able to use the new driver we need to add its name to the list of enabled drivers in `oned.conf`:

```
AUTH_MAD = [
    executable = "one_auth_mad",
    authn = "ssh,x509,ldap,server_cipher,server_x509,length"
]
```

4.7 Cloud Bursting Driver

This guide will show you how to develop a new driver for OpenNebula to interact with an external cloud provider.

4.7.1 Overview

Cloud bursting is a model in which the local resources of a Private Cloud are combined with resources from remote Cloud providers. The remote provider could be a commercial Cloud service, such as Amazon EC2, or a partner infrastructure running a different OpenNebula instance. Such support for cloud bursting enables highly scalable hosting environments. For more information on this model see the *Cloud Bursting overview*

The remote cloud provider will be included in the OpenNebula host pool like any other physical host of your infrastructure:

```
$ onehost create remote_provider im_provider vmm_provider tm_dummy dummy

$ onehost list
  ID NAME            CLUSTER   RVM      ALLOCATED_CPU      ALLOCATED_MEM STAT
   2 kvm-            -           0       0 / 800 (0%)      0K / 16G (0%) on
   3 kvm-1           -           0       0 / 100 (0%)     0K / 1.8G (0%) on
   4 remote_provider -           0       0 / 500 (0%)     0K / 8.5G (0%) on
```

When you create a new host in OpenNebula you have to specify the following parameters:

- Name: remote_provider

Name of the host, in case of physical hosts it will the ip address or hostname of the host. In case of remote providers it will be just a name to identify the provider.

- *Information Manager*: `im_provider`

IM driver gather information about the physical host and hypervisor status, so the OpenNebula scheduler knows the available resources and can deploy the virtual machines accordingly.

- *VirtualMachine Manager*: `vmm_provider`

VMM drivers translate the high-level OpenNebula virtual machine life-cycle management actions, like deploy, shutdown, etc. into specific hypervisor operations. For instance, the KVM driver will issue a virsh create command in the physical host. The EC2 driver translate the actions into Amazon EC2 API calls.

- *Transfer Manager*: `tm_dummy`

TM drivers are used to transfer, clone and remove Virtual Machines Image files. They take care of the file transfer from the OpenNebula image repository to the physical hosts. There are specific drivers for different storage configurations: shared, non-shared, lvm storage, etc.

- *VirtualNetwork Manager*: `dummy`

VNM drivers are used to set the network configuration in the host (firewall, 802.1Q, ebtables, osvswitch)

When creating a new host to interact with a remote cloud provider we will use mock versions for the TM and VNM drivers. Therefore, we will only implement the functionality required for the IM and VMM driver.

4.7.2 Adding the Information Manager

Edit oned.conf

Add a new IM section for the new driver in oned.conf:

```
#*******************************************************************************
# Information Driver Configuration
#*******************************************************************************
# You can add more information managers with different configurations but make
# sure it has different names.
#
#   name      : name for this information manager
#
#   executable: path of the information driver executable, can be an
#               absolute path or relative to $ONE_LOCATION/lib/mads (or
#               /usr/lib/one/mads/ if OpenNebula was installed in /)
#
#   arguments : for the driver executable, usually a probe configuration file,
#               can be an absolute path or relative to $ONE_LOCATION/etc (or
#               /etc/one/ if OpenNebula was installed in /)
#    -r number of retries when monitoring a host
#    -t number of threads, i.e. number of hosts monitored at the same time
#*******************************************************************************
#-------------------------------------------------------------------------------
#  EC2 Information Driver Manager Configuration
#-------------------------------------------------------------------------------
IM_MAD = [
      name       = "im_provider",
      executable = "one_im_sh",
      arguments  = "-t 1 -r 0 provider_name" ]
#-------------------------------------------------------------------------------
```

Populating the Probes

Create a new directory to store your probes, the name of this folder must match the name provided in the arguments section of the IM_MAD in oned.conf:

- /var/lib/one/remotes/im/<provider_name>.d

These probes must return:

- *Information of the host capacity*, to limit the number of VMs that can be deployed in this hosts.
- *Information of the VMs* running in this host-

You can see an example of these probes in the ec2 driver (code) included in OpenNebula

You must include the PUBLIC_CLOUD and HYPERVISOR attributes as one of the values returned by your probes, otherwise OpenNebula will consider this host as local. The HYPERVISOR attribute will be used by the scheduler and should match the TYPE value inside the PUBLIC_CLOUD section provided in the VM template.

```
PUBLIC_CLOUD="YES"
HYPERVISOR="provider_name"
```

4.7.3 Adding the Virtual Machine Manager

Edit oned.conf

```
#*******************************************************************************
# Virtualization Driver Configuration
#*******************************************************************************
# You can add more virtualization managers with different configurations but
# make sure it has different names.
#
#   name      : name of the virtual machine manager driver
#
#   executable: path of the virtualization driver executable, can be an
#               absolute path or relative to $ONE_LOCATION/lib/mads (or
#               /usr/lib/one/mads/ if OpenNebula was installed in /)
#
#   arguments : for the driver executable
#    -r number of retries when monitoring a host
#    -t number of threads, i.e. number of hosts monitored at the same time
#
#   default   : default values and configuration parameters for the driver, can
#               be an absolute path or relative to $ONE_LOCATION/etc (or
#               /etc/one/ if OpenNebula was installed in /)
#
#   type      : driver type, supported drivers: xen, kvm, xml
#-------------------------------------------------------------------------------
VM_MAD = [
    name       = "vmm_provider",
    executable = "one_vmm_sh",
    arguments  = "-t 15 -r 0 provider_name",
    type       = "xml" ]
#-------------------------------------------------------------------------------
```

Create the Driver Folder and Implement the Specific Actions

Create a new folder inside the remotes dir (/var/lib/one/remotes/vmm). The new folder should be named "providet_name", the name specified in the previous VM_MAD arguments section.

This folder must contain scripts for the supported actions. You can see the list of available actions in the *Virtual Machine Driver guide*. These scripts are language-agnostic so you can implement them using python, ruby, bash...

You can see examples on how to implement this in the ec2 driver:

- EC2 Shutdown action:

```
#!/usr/bin/env ruby

# ---------------------------------------------------------------------------- #
```

```
# Copyright 2010-2013, C12G Labs S.L                                          #
#                                                                             #
# Licensed under the Apache License, Version 2.0 (the "License"); you may    #
# not use this file except in compliance with the License. You may obtain    #
# a copy of the License at                                                    #
#                                                                             #
# http://www.apache.org/licenses/LICENSE-2.0                                  #
#                                                                             #
# Unless required by applicable law or agreed to in writing, software        #
# distributed under the License is distributed on an "AS IS" BASIS,           #
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.    #
# See the License for the specific language governing permissions and         #
# limitations under the License.                                              #
# --------------------------------------------------------------------------- #

$: << File.dirname(__FILE__)

require 'ec2_driver'

deploy_id = ARGV[0]
host      = ARGV[1]

ec2_drv = EC2Driver.new(host)

ec2_drv.shutdown(deploy_id)
```

Create the New Host

After restarting oned we can create the new host that will use this new driver

```
$ onehost create remote_provider im_provider vmm_provider tm_dummy dummy
```

Create a new Virtual Machine

Create a new VM using a template with an specific section for this provider. You have to include the required information to start a new VM inside the PUBLIC_CLOUD section, and the TYPE attribute must match the HYPERVISOR value of the host. For example:

```
$ cat vm_template.one
CPU=1
MEMORY=256
PUBLIC_CLOUD=[
    TYPE=provider_name
    PROVIDER_IMAGE_ID=id-141234,
    PROVIDER_INSTANCE_TYPE=small_256mb
]

$ onevm create vm_template
ID: 23

$ onevm deploy 23 remote_provider
```

After this, the deploy script will receive the following arguments:

- The path to the deployment file that contains the following XML:

```
<CPU>1</CPU>
<MEMORY>256</MEMORY>
<PUBLIC_CLOUD>
    <TYPE>provider_name</TYPE>
    <PROVIDER_IMAGE_ID>id-141234</PROVIDER_IMAGE_ID>
    <PROVIDER_INSTANCE_TYPE>small_256mb</PROVIDER_INSTANCE_TYPE>
</PUBLIC_CLOUD>
```

- The hostname: `remote_provider`
- The VM ID: `23`

The deploy script has to return the ID of the new resource and an exit_code 0:

```
$ cat /var/lib/one/remote/provider/deploy
#!/bin/bash
deployment_file=$1
# Parse required parameters from the template
..
# Retrieve account credentials from a local file/env
...
# Create a new resource using the API provider
...
# Return the provider ID of the new resource and exit code 0 or an error message
```

CHAPTER

FIVE

REFERENCES

5.1 Custom Routes for Sunstone Server

OpenNebula Sunstone server plugins consist a set files defining custom routes. Custom routes will have priority over default routes and allow administrators to integrate their own custom controllers in the Sunstone Server.

5.1.1 Configuring Sunstone Server Plugins

It is very easy to enable custom plugins:

1. Place your custom routes in the `/usr/lib/one/sunstone/routes` folder.
2. Modify `/etc/one/sunstone-server.conf` to indicate which files should be loaded, as shown in the following example:

```
# This will load ''custom.rb'' and ''other.rb'' plugin files.
:routes:
    - custom
    - other
```

5.1.2 Creating Sunstone Server Plugins

Sunstone server is a Sinatra application. A server plugin is simply a file containing one or several custom routes, as defined in sinatra applications.

The following example defines 4 custom routes:

```
get '/myplugin/myresource/:id' do
    resource_id = params[:id]
    # code...
end

post '/myplugin/myresource' do
    # code
end

put '/myplugin/myresource/:id' do
    # code
end

del '/myplugin/myresource/:id' do
```

```
    # code
end
```

Custom routes take preference over Sunstone server routes. In order to easy debugging and ensure that plugins are not interfering with each others, we recommend however to place the routes in a custom namespace (`myplugin` in the example).

From the plugin code routes, there is access to all the variables, helpers etc which are defined in the main sunstone application code. For example:

```
opennebula_client = $cloud_auth.client(session[:user])
sunstone_config = $conf
logger.info("New route")
vm3_log = @SunstoneServer.get_vm_log(3)
```

5.2 Building from Source Code

This page will show you how to compile and install OpenNebula from the sources.

If you want to install it from your package manager, visit the software menu to find out if OpenNebula is included in your official distribution package repositories.

Warning: Do not forget to check the *Building Dependecies* for a list of specific software requirements to build OpenNebula.

5.2.1 Compiling the Software

Follow these simple steps to install the OpenNebula software:

- Download and untar the OpenNebula tarball.
- Change to the created folder and run scons to compile OpenNebula

```
$ scons [OPTION=VALUE]
```

```
the argument expression [OPTION=VALUE] is used to set non-default values for :
```

OPTION	VALUE
syslog	**yes** to compile syslog support. Needs log4cpp lib
sqlite_db	path-to-sqlite-install
sqlite	**no** if you don't want to build sqlite support
mysql	**yes** if you want to build mysql support
xmlrpc	path-to-xmlrpc-install
parsers	**yes** if you want to rebuild flex/bison files
new_xmlrpc	**yes** if you have an xmlrpc-c version >= 1.31

If the following error appears, then you need to remove the option 'new_xmlrpc=yes' or install xmlrpc-c version >= 1.31:

```
error: 'class xmlrpc_c::serverAbyss::constrOpt' has no member named 'maxConn'
```

- OpenNebula can be installed in two modes: `system-wide`, or in `self-contained` directory. In either case, you do not need to run OpenNebula as root. These options can be specified when running the install script:

```
./install.sh <install_options>
```

where <*install_options*> can be one or more of:

OPTION	VALUE
-u	user that will run OpenNebula, defaults to user executing install.sh
-g	group of the user that will run OpenNebula, defaults to user executing install.sh
-k	keep configuration files of existing OpenNebula installation, useful when upgrading. This flag should not be set when installing OpenNebula for the first time
-d	target installation directory. If defined, it will specified the path for the **self-contained** install. If not defined, the installation will be performed **system wide**
-c	only install client utilities: OpenNebula cli, occi and ec2 client files
-r	remove Opennebula, only useful if -d was not specified, otherwise `rm -rf $ONE_LOCATION` would do the job
-h	prints installer help

The packages do a `system-wide` installation. To create a similar environment, create a `oneadmin` user and group, and execute:

```
oneadmin@frontend:~/ $> wget <opennebula tar gz>
oneadmin@frontend:~/ $> tar xzf <opennebula tar gz>
oneadmin@frontend:~/ $> cd one-4.0
oneadmin@frontend:~/one-4.0/ $> scons -j2 mysql=yes syslog=yes
[ lots of compiling information ]
scons: done building targets.
oneadmin@frontend:~/one-4.0 $> sudo ./install.sh -u oneadmin -g oneadmin
```

5.2.2 Ruby Dependencies

Ruby version needs to be:

- **ruby** >= 1.8.7

Some OpenNebula components need ruby libraries. Some of these libraries are interfaces to binary libraries and the development packages should be installed in your machine. This is the list of the ruby libraries that need a development package:

- **sqlite3**: sqlite3 development library
- **mysql**: mysql client development library
- **curb**: curl development library
- **nokogiri**: expat development librarie
- **xmlparse**: libxml2 and libxslt development libraries

You will also need ruby development package to be able to compile these gems.

We provide a script to ease the installation of these gems. it is located in `/usr/share/one/install_gems` (system-wide mode) or `$ONE_LOCATION/share/install_gems` (self-contained mode). It can be called with the components you want the gem dependencies to be installed. Here are the options:

- **optional**: libraries that make CLI and OCA faster
- **quota**: quota system
- **sunstone**: sunstone graphical interface
- **cloud**: ec2 and occi interfaces

- **ozones_client**: CLI of ozones
- **ozones_server**: server part of ozones, both mysql and sqlite support
- **ozones_server_sqlite**: ozones server, only sqlite support
- **ozones_server_mysql**: ozones server, only mysql support
- **acct**: accounting collector, both mysql and sqlite support
- **acct_sqlite**: accounting collector, only sqlite support
- **acct_mysql**: accounting collector, only mysql support

The tool can be also called without parameters and all the packages will be installed.

For example, to install only requirements for sunstone, ec2 and occi interfaces you'll issue:

```
oneadmin@frontend: $> ./install_gems sunstone cloud
```

5.3 Build Dependencies

This page lists the **build** dependencies for OpenNebula.

If you want to install it from your package manager, visit the software menu to find out if OpenNebula is included in your official distribution package repositories.

- **g++** compiler (>= 4.0)
- **xmlrpc-c** development libraries (>= 1.06)
- **scons** build tool (>= 0.98)
- **sqlite3** development libraries (if compiling with sqlite support) (>= 3.6)
- **mysql** client development libraries (if compiling with mysql support) (>= 5.1)
- **log4cpp** flexible logging library (if compiling with syslog support) (>=1.0)
- **libxml2** development libraries (>= 2.7)
- **openssl** development libraries (>= 0.9.8)
- **ruby** interpreter (>= 1.8.7)

5.3.1 Debian/Ubuntu

- **g++**
- **libxmlrpc-c3-dev**
- **scons**
- **libsqlite3-dev**
- **libmysqlclient-dev**
- **libxml2-dev**
- **libssl-dev**
- **liblog4cpp5-dev**
- **ruby**

5.3.2 CentOS 6

- **gcc-c++**
- **libcurl-devel**
- **libxml2-devel**
- **xmlrpc-c-devel**
- **openssl-devel**
- **mysql-devel**
- **log4cpp-devel**
- **openssh**
- **pkgconfig**
- **ruby**
- **scons**
- **sqlite-devel**
- **xmlrpc-c**
- **java-1.7.0-openjdk-devel**

5.3.3 CentOS 5 / RHEL 5

scons

The version that comes with Centos is not compatible with our build scripts. To install a more recent version you can download the RPM at:

http://www.scons.org/download.php

```
$ wget http://prdownloads.sourceforge.net/scons/scons-1.2.0-1.noarch.rpm
$ yum localinstall scons-1.2.0-1.noarch.rpm
```

xmlrpc-c

You can download the xmlrpc-c and xmlrpc-c packages from the rpm repository at http://centos.karan.org/.

```
$ wget http://centos.karan.org/el5/extras/testing/i386/RPMS/xmlrpc-c-1.06.18-1.el5.kb.i386.rpm
$ wget http://centos.karan.org/el5/extras/testing/i386/RPMS/xmlrpc-c-devel-1.06.18-1.el5.kb.i386.rpm
$ yum localinstall --nogpgcheck xmlrpc-c-1.06.18-1.el5.kb.i386.rpm xmlrpc-c-devel-1.06.18-1.el5.kb.i
```

sqlite

This package should be installed from source, you can download the tar.gz from http://www.sqlite.org/download.html. It was tested with sqlite 3.5.9.

```
$ wget http://www.sqlite.org/sqlite-amalgamation-3.6.17.tar.gz
$ tar xvzf sqlite-amalgamation-3.6.17.tar.gz
$ cd sqlite-3.6.17/
$ ./configure
```

```
$ make
$ make install
```

If you do not install it to a system wide location (/usr or /usr/local) you need to add LD_LIBRARY_PATH and tell scons where to find the files:

```
$ scons sqlite=<path where you installed sqlite>
```

Ruby

Ruby package is needed during install process

```
$ yum install ruby
```

5.3.4 openSUSE 11.3

Building tools

By default openSUSE 11 does not include the standard building tools, so before any compilation is done you should install:

```
$ zypper install gcc gcc-c++ make patch
```

Required Libraries

Install these packages to satisfy all the dependencies of OpenNebula:

```
$ zypper install libopenssl-devel libcurl-devel scons pkg-config sqlite3-devel libxslt-devel libxmlrp
```

Ruby

We can install the standard packages directly with zypper:

```
$ zypper install ruby ruby-doc-ri ruby-doc-html ruby-devel rubygems
```

rubygems must be >=1.3.1, so to play it safe you can update it to the latest version:

```
$ wget http://rubyforge.org/frs/download.php/45905/rubygems-1.3.1.tgz
$ tar zxvf rubygems-1.3.1.tgz
$ cd rubygems-1.3.1
$ ruby setup.rb
$ gem update --system
```

Once rubygems is installed we can install the following gems:

```
gem install nokogiri rake xmlparser
```

xmlrpc-c

xmlrpc-c must be built by downloading the latest svn release and compiling it. Read the README file included with the package for additional information.

```
svn co http://xmlrpc-c.svn.sourceforge.net/svnroot/xmlrpc-c/super_stable xmlrpc-c
cd xmlrpc-c
./configure
make
make install
```

5.3.5 MAC OSX 10.4 10.5

OpenNebula frontend can be installed in Mac OS X. Here are the dependencies to build it in 10.5 (Leopard)

Requisites:

- **xcode** (you can install in from your Mac OS X DVD)
- **macports** http://www.macports.org/

Getopt

This package is needed as `getopt` that comes with is BSD style.

```
$ sudo port install getopt
```

xmlrpc

```
$ sudo port install xmlrpc-c
```

scons

You can install scons using macports as this:

```
$ sudo port install scons
```

Unfortunately it will also compile python an lost of other packages. Another way of getting it is downloading the standalone package in http://www.scons.org/download.php. Look for scons-local Packages and download the Gzip tar file. In this example I am using version 1.2.0 of the package.

```
$ mkdir -p ~/tmp/scons
$ cd ~/tmp/scons
$ tar xvf ~/Downloads/scons-local-1.2.0.tar
$ alias scons='python ~/tmp/scons/scons.py'
```

5.3.6 Gentoo

When installing libxmlrpc you have to specify that it will be compiled with thread support:

```
# USE="threads" emerge xmlrpc-c
```

5.3.7 Arch

They are listed in this PKGBUILD.