Current IaaS API Data Schemas

Foreword

This is draft version of this IaaS API model review. There will be further updates and revisions of this document, especially around the areas of storage and networking. There maybe links that are inaccessible if you do not have SLA@SOI project membership, but if you have any queries in respect to the content pointed to by those links please contact Andy Edmonds (andrewx.edmonds@intel.com).

Scope

Any infrastructure as a service provider that publishes a schema (data model) that allow the creation of compute resources. Storage and networking aspects are not reviewed for now.

General Requirements

- Keep the structure of a resource’s representation as simple as possible
- Support the ability to represent more than just virtual machines (e.g. physical machines, routers, load balancers etc) using the same schema.
- The model should be extendable so that entities within can be extended with additional provider-specific parameters.
- Do not "leak" technology into the representation (e.g. dependence on IPv4 or IPv6).
- Keep the resource representation’s hierarchy depth to a minimum.
- Do seperate functional attributes from non-functional attributes via grouping/categorisation.
- Representation model should be equal or a super set of the request model;
    - e.g. I request a resource to have 1GB of RAM. When I view its representation after it is provisioned I see that it has the attribute representing 1GB of RAM but also that is has other attributes such as hostname.
    - This allows for the reuse of the model within the provider’s infrastructure landscape and promotes a common "language" and reuse.
- Provide multiple renderings of the representation e.g. XML, JSON, CSV, RDF. Whereever possible these renderings should conform to a schema. Wrapping these renderings inside another schema (e.g. SOAP) should be avoided if possible. This reduces complexity by removing additional dependencies.
    - OVF would be an excellent candidate to provide a standardised rendering of resources, however it is unclear how it can support attributes that are not in its schema TODO.

Amazon EC2

API: API Review

Resources available: Amazon EC2 provides compute resources. One resource is known as an Instance.

Supported Representation Rendering: XML conforming to WSDL Schema

As Amazon EC2 is an RPC driven API the follow entites of the model are passed to methods as individual parameters as opposed to a document containing all parameters to the method.

Compute Resource Request

The majority of the functional requirements of infrastructure contained in an EC2 request are embedded in the instanceType parameter. Further to those functional requirements are imageId...

`imageId, minCount, maxCount, keyName, groupSet, userData, instanceType, placement, kernelId, ramdiskId, blockDeviceMapping`

Compute Resource Representation

This model is returned when clients query for information about what instances are currently running.

```
amiLaunchIndex, dnsName, imageId, instanceId, instanceState,
instanceType, keyName, kernelId, launchTime, placement, privateDnsName,
productCodes, ramdiskId, reason
```

Sun Cloud API

API: API Review

Resources available: Sun provides compute resources. One resource is known as a virtual machine (VM).

Supported Representation Rendering: JSON

The Sun Cloud API is consitent in it's request and resource representation models. A request is a subset of a resource's representation. This is as it uses the REST approach as it's underlying architectural principle. In order for a new resource to be created, all one needs to do is to supply a VM model to the URI that allows the creation of VMs.

Compute Resource Request

See compute resource representation.

Compute Resource Representation

Field Name	Type	Occurs	Description
name	String	1	Logical name of this VM, unique within the owning Cluster. [POST][PUT]
uri	URI	1	GET, PUT, and DELETE may be used to retrieve representations of the VM, to update it, and to delete it, respectively.
model_status	String	0..1	Current modeling status of this virtual machine (read only). FIXME - enumerate valid values beyond "deployed", "error", etc.

run_status	String	0..1	Current running status of this virtual machine (read only). FIXME - enumerate valid values beyond "halted", "running", "paused".
from_template	String	0..1	On a *Create VM* request type, the URI of an existing *VM template* to initialize default values from. [POST]
from_vm	String	0..1	On a *Create VM* request type, the URI of an existing *VM* to initialize default values from. [POST]
description	String	0..1	Human friendly description of this virtual machine. [POST][PUT]
hostname	String	0..1	Fully qualified host name of this virtual machine.
os	String	0..1	Operating system running on the VM being modelled. FIXME - enumerate the legal values. [POST]
cpu	Integer	0..1	CPU speed in Mhz. [POST]
memory	Integer	0..1	Main memory size in GB. [POST]
boot_disk	Integer	0..1	Boot disk space to allocate in GB. [POST]
data_disk	Integer	0..1	Data disk space to allocate in GB. [POST]
temp_disk	Integer	0..1	Temporary disk space to allocate in GB. [POST]
params	{ }	0..1	Configuration parameters for this VM, keyed by parameter name. The list of system defined configuration parameters is TBD, but one of them will be "user_params", whose value is a hash of arbitrary user defined configuration parameters. [POST][PUT]
tags	[ ]	0..1	Arbitrary values assigned by the user. These values have no semantic impact on the operation of the cloud. [POST][PUT]
back_up	URI	1	A POST of a Backup representation to this URI requests creation of a new Backup.
attach	URI	1	A POST of a PublicAddress representation to this URI requests connection to a Public Address. Or, a POST of a VNet representation to this URI requests connection to a VNet.
detach	URI	1	A POST of a PublicAddress representation to this URI requests disconnection from a Public Address. Or, a POST of a VNet representation to this URI reequests disconnection from a VNet.
backups	Backup[]	0..1	Backup snapshots from this virtual machine.
interfaces	Interface[]	1	Network interfaces associated with this virtual machine. These are created automatically when VNets or Public Addresses are associated with VM.

controllers	{}	0..1	Hash of URIs which may be used to request state changes in the VM via POST requests (see next table), keyed by the type of state change being requested.

Flexiscale

API: API Review

Resources available: Flexiscale provides virtual compute resources. One resource is known as a server.

Supported Representation Rendering: XML conforming to WSDL Schema

Compute Resource Request

In order for a new server to be created, all one needs to do is to supply a server model to the URI that allows the creation of servers. See Compute Resource Representation.

Compute Resource Representation

```
<xsd:complexType name="Server">
        <xsd:all>
                <xsd:element name="server_id" type="xsd:int" minOccurs="0"/>
                <xsd:element name="server_name" type="xsd:string" minOccurs="0"/>
                <xsd:element name="status" type="xsd:string" minOccurs="0"/>
                <xsd:element name="package_id" type="xsd:int" minOccurs="0"/>
                <xsd:element name="processors" type="xsd:int" minOccurs="0"/>
                <xsd:element name="memory" type="xsd:int" minOccurs="0"/>
                <xsd:element name="operating_system_image" type="typens:OperatingSystemImage" minOccurs="0"/>
                <xsd:element name="disk_capacity" type="xsd:long" minOccurs="0"/>
                <xsd:element name="disks" type="typens:ArrayOf_xsd_int" minOccurs="0"/>
                <xsd:element name="network_interfaces" type="typens:ArrayOf_xsd_int" minOccurs="0"/>
                <xsd:element name="initial_password" type="soapenc:string" minOccurs="0"/>
                <xsd:element name="uptime" type="xsd:long" minOccurs="0"/>
                <xsd:element name="ip_addresses" type="typens:ArrayOf_xsd_string" minOccurs="0"/>
                <xsd:element name="modified" type="xsd:boolean" minOccurs="0"/>
        </xsd:all>
</xsd:complexType>
```

ElasticHosts

API: API Review

Resources available: ElasticHosts provides virtual compute resources. One resource is known as a server.

Supported Representation Rendering: JSON, CSV

Compute Resource Request

In order for a new server to be created, all one needs to do is to supply a server model to the URI that allows the creation of servers. See Compute Resource Representation.

Compute Resource Representation

Key	Value
name	Server name.
parent	UUID of parent object, if any.
cpu	cpu quota in core MHz.
smp	number of virtual processors or ’auto’ to calculate based on cpu.
mem	virtual memory size in MB.
ide:BUS[0-1]:UNIT[0-1] **scsi:BUS[0]:UNIT[0-7]** **block:INDEX[0-7]**	Drive UUID to connect as specified device.
ide:BUS[0-1]:UNIT[0-1]:media **scsi:BUS[0]:UNIT[0-7]:media** **block:INDEX[0-7]:media**	Media type - set to ’cdrom’ to simulate a cdrom, set to ’disk’ or leave unset to simulate a hard disk.
boot	Device to boot, e.g. ide:0:0 or ide:1:0. Set to a list to make multiple devices bootable.
nic:0:model	Create network interface with given type (use ’e1000’ as default value).
nic:0:dhcp	The IP address offered by DHCP to network interface 0. If unset, no address is offered. Set to ’auto’ to allocate a temporary IP at boot.
vnc:ip	IP address for overlay VNC access on port 5900. Set vnc:ip=’auto’, to reuse nic:0:dhcp if available, or otherwise allocate a temporary IP at boot.
vnc:password	Password for VNC access
vnc:tls	Set to ’on’ to require VeNCrypt-style TLS auth in addition to the password. If this is unset, only unencrypted VNC is available.

GoGrid

API: API Review

Resources available: ElasticHosts provides virtual compute resources. One resource is known as a server.

Supported Representation Rendering: XML, JSON, CSV

Compute Resource Request

To create a server on GoGrid the following is supplied as a name-value tuple:

- api_key
- sig
- v
- name
- image
- ram
- ip
- description (opt)
- format (opt)

Compute Resource Representation

A server on the GoGrid is represented by a Server entity. A response containing server representations will be a list of servers along with summary containing total; the total number returned, start; the start index of the list (always 0) and a variable named returned which is equal to total. The attributes returned for server in the list are:

- id
- name
- description
- ip
- image
- ram
- state
- type
- os

Enomalism

API:API Review

Resources available: Enomalism provides virtual compute resources. One resource is known as a virtual machine.

Supported Representation Rendering: XML

Large dependence on Libvirt. Below is how machines are tracked (the landscape) within enomaly. What’s shown is the a table of the enomaly system, in which machines running are stored. There are a number of attributes but most interesting is the "xmldesc" field. This field corresponds to the Libvirt XML descriptor of the virtual machine.

```
 CREATE TABLE ‘machine‘ (
  ‘id‘ int(11) NOT NULL auto_increment,
  ‘ip_addr‘ text,
  ‘ip_port‘ int(11) default NULL,
  ‘rest_baseurl‘ text,
  ‘uuid‘ varchar(36) NOT NULL,
  ‘parent_id‘ int(11) default NULL,
  ‘suspend_data‘ varchar(512) default NULL,
  ‘machine_name‘ varchar(512) NOT NULL,
  ‘xmldesc‘ text,
  PRIMARY KEY  (‘id‘),
  UNIQUE KEY ‘uuid‘ (‘uuid‘),
  UNIQUE KEY ‘machine_name‘ (‘machine_name‘),
  UNIQUE KEY ‘suspend_data‘ (‘suspend_data‘),
  KEY ‘machine_parent_id_exists‘ (‘parent_id‘)
)
```

Compute Resource Request

Virtual machines in Enomalism are invariably requested using the Libvirt XML descriptor. In order to create a virtual machine in enomaly a number of calls are required until one is up and running. This implies a certain amount of state is maintained through out the calls that create the VM. This is counter to REST principles.

Compute Resource Representation

Are represented using the Libvirt XML descriptor.

OpenNebula

API: API Review

Resources available: OpenNebula provides virtual compute resources. One resource is known as a virtual machine (VM).

Supported Representation Rendering: Unclear

Compute Resource Request

To request the creation of a VM, a user has to supply a VIrtual Machine Template string to the one.vmallocate RPC method. This string is a vector of name-value pairs. This is convienent as it allows easy extension. The current set of attributes are:

Name	Description
NAME	Name that the VM will get, for description purposes. If **NAME** is not supplied a name generated by one will be in the form of one-`vm_id`.
CPU	Percentage of CPU divided by 100 required for the Virtual Machine. Half a processor is written 0.5.
MEMORY	Amount of RAM required for the VM, in Megabytes.
DISK	Image file to be used to deploy the Virtual Machine. Disk attributes: - **IMAGE**: Path of the image file. - **DEVICE**: Device associated to the image file.
BOOT	Tells the kernel what device it is going to boot from.
KERNEL	The kernel file that will be used to boot the VM's, normally located at /boot (i.e. /boot/vmlinuz-2.6.XX.X-xen).
RAMDISK	The initial ramdisk file that will be used to boot the VM's, normally located at /boot (i.e. /boot/initrd.img-2.6.XX.X-xen).
NIC	Network interface, attributes: - **MAC**. HW address associated with the network interface. - **BRIDGE**: Name of the bridge the network device is going to be attached to.
REQUIREMENTS	Boolean expression that rules out provisioning hosts from list of machines suitable to run this VM.
RANK	This field sets which attribute will be used to sort the suitable hosts for this VM. Basically, it defines which hosts are *more suitable* than others.

What's also useful with this type of request representation is that logical operators can be used to narrow resource selection (e.g. CPUSPEED>1000) via the REQUIREMENTS vector name.

Compute Resource Representation

A representation of a VM can be retreived by issuing the command onevm show <vm_id>. This will return back a VM representation containing the following information:

ID	ONE VM identifier
NAME	Name of the ONE
STAT	Status of the VM
CPU	CPU percentage used by the VM
MEM	Memory used by the VM
HOSTNAME	Host where the VM is being or was run
TIME	Time since the submission of the VM (days hours:minutes:seconds

Slicehost

API: API Review

Resources available: Slicehost provides virtual compute resources. One resource is known as a slice.

Supported Representation Rendering: XML

Compute Resource Request

Typically it is enough to specify a slice’s image, flavour and name (see below) when requesting a new provisioning.

Compute Resource Representation

Field	Access	Notes
name	rw	A string to identify the host
flavor_id	rw	Flavor - name, price and amount of RAM available for this slice offering (slices increase by 256MB RAM intervals)
image_id	rw	Image - name: the name of the OS image to run
backup_id	rw	Backup - a pointer to a backup image of the slice its name and when it was last backed up
status	w	Current status of the slice
progress	w	Progress of the current action
bw_in	w	Total incoming bandwidth per Gigbyte for the slice in the current billing cycle
bw_out	w	Same as above but for outgoing
addresses	w	Array of IP addresses assigned to the slice
ip_address	w	Deprecated
root_password	w	Shown only on creation of slice

Eucalyptus

See Amazon EC2

Globus Nimbus

See Amazon EC2

AppNexus

API Review

As AppNexus do not offer an easily accessible programmable API (only command line tools), and so is not included in this review.

F5.com

Compute Resource Request

Compute Resource Representation

Apache Tashi

API: API Review

Resources available: Tashi provides virtual compute resources. One resource is known as an Instance.

Supported Representation Rendering: Unclear

Compute Resource Request

To create an instance using Tashi, a call to the cluster manager interface is made. This is done calling the createVm method and passing to it a Instance object (see representation below).

Compute Resource Representation

Apache Tashi uses Thrift to model Instances. The Thrift structure that represents an instance is as follows:

```
struct Instance {
        1:i32 id,
        2:i32 vmId,
        3:i32 hostId,
        4:bool decayed,
        5:InstanceState state,
        6:i32 userId,
        7:string name, // User specified
        8:i32 cores, // User specified
        9:i32 memory, // User specified
        10:list<DiskConfiguration> disks, // User specified
        11:list<NetworkConfiguration> nics // User specified
        12:map<string, string> hints // User specified
}
```

To represent disk and network configuration the following structs are used:

```
struct DiskConfiguration {      1:string uri,   2:bool persistent}struct NetworkConfiguration { 1:i32 network   2:string mac}
```

CohesiveFT

Compute Resource Request

Compute Resource Representation

Attached Files

Attached Images