The Cloud Infrastructure Management Interface (CIMI)

July 2013

Agenda

- What problems does CIMI Solve?
- What are the benefits of CIMI?
- CMWG Background & Status
- CIMI Model
- CIMI REST/HTTP-Based Protocol
- Questions

What problems does CIMI Solve?

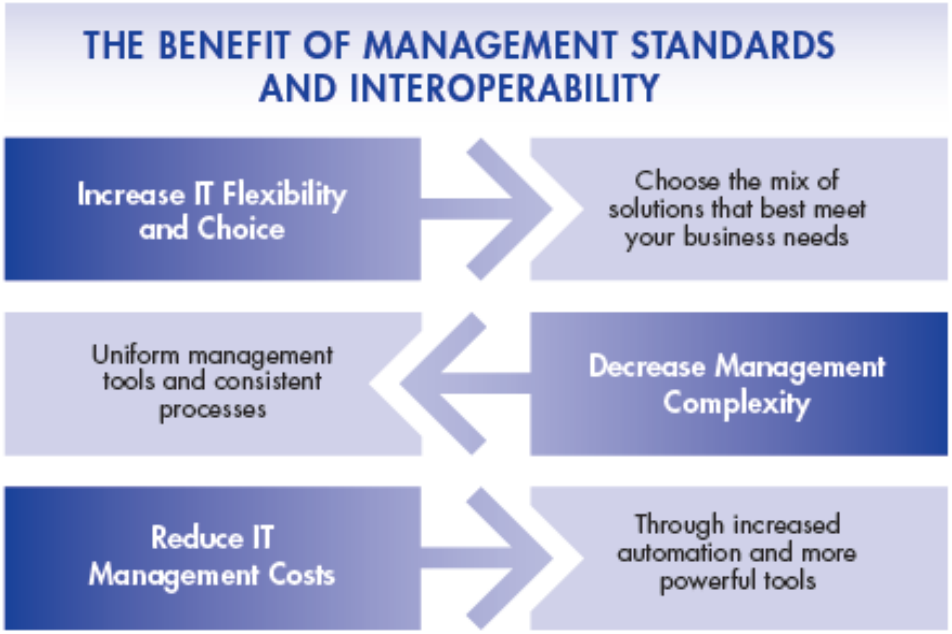
- Cloud customers are using **various interfaces** to manage their public and private clouds:
  - EC2, OpenStack Nova, Cloud Stack, Open Nebula, vendor specific
- Each API **involves work to develop**, **test** and **maintain**, continually evolving
  - Little to no stability, versioning support, or backward compatibility guarantees
- APIs are **under control of specific vendors**, not open standards organizations
- Open Source projects (CloudStack, OpenStack, Eucalyptus) only interoperate if everybody is **using the same code** – no winners here
- Customers need multiple clouds to balance risk and so they must either use only clouds with the same code, or **write multiple adapters** to each cloud

What are the benefits of CIMI?

- *CIMI is a stable, open, standard implemented on various open source projects and is under change control of an open, international standards body (DMTF)*
- Multiple Clouds with a Common Interface
  - CIMI reduces the effort to use multiple clouds via an interoperable API
  - The reduced friction between clouds will grow the overall market for cloud computing
- Cloud Standards Adoption
  - CIMI was created by a large collaboration of cloud vendors, service providers and telecom companies
  - Existing cloud APIs were submitted to DMTF to start the work
  - Experience with real cloud issues has been factored into the standard
- Cloud Offerings can continue to innovate
  - CIMI is extensible for both unique vendor features as well as future versions of the standard
  - CIMI supports OVF™ for Cloud Portability

The Importance of Cloud Standards

With the ever-increasing need for flexibility, availability and performance in today's distributed enterprises, management standards for IT professionals are now more important than ever.



Deploying systems, tools and solutions that support management standards helps reduce system management complexity, lower costs and improve agility.

CIMI Background

Cloud Incubator (2009-2010) published informational specifications

- "Interoperable Clouds" white paper
- Architecture and interfaces
- Use cases and resource interaction model

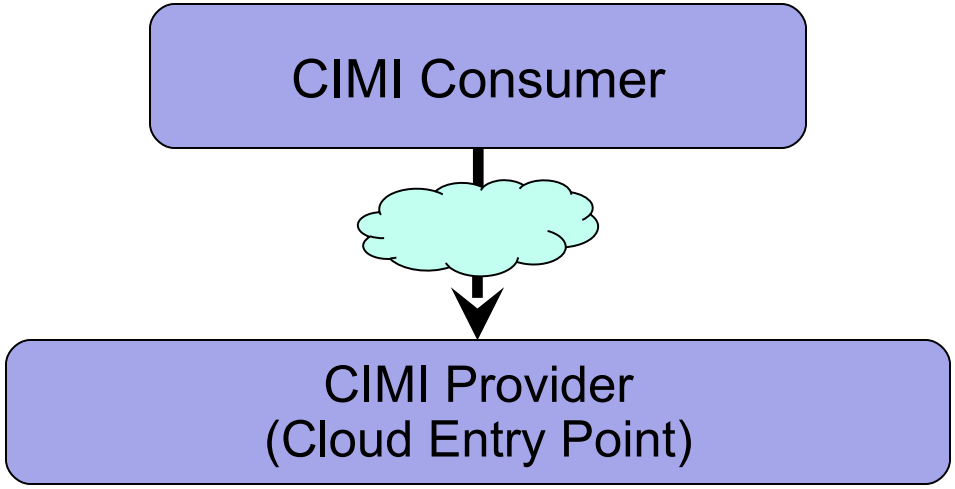
Cloud Management Working Group (CMWG) started July 2010

- Scope: IaaS
- Chartered Deliverables:
  - IaaS Model, Requirements,
  - HTTP REST-based,
  - Other notes/whitepapers (e.g. Primers)
- Leverage other standards, e.g., OVF (Open Virtualization Format) , CIM
- 40+ actively involved companies, academic and alliance members
- Many input submissions... work started fresh

CIMI Model Essential Characteristics

- **Core IaaS functionality**
  - Deploying, monitoring and managing:
    - Machines, Volumes, and Networks
  - Supporting Client Roles:
    - VM/application developer, deployer and administrator
  - Supporting Server Roles:
    - IaaS Cloud Provider
- Enables mapping to existing IaaS models in the community
- Simplicity and flexibility

CIMI Model – Getting Started



Cloud Entry Point:

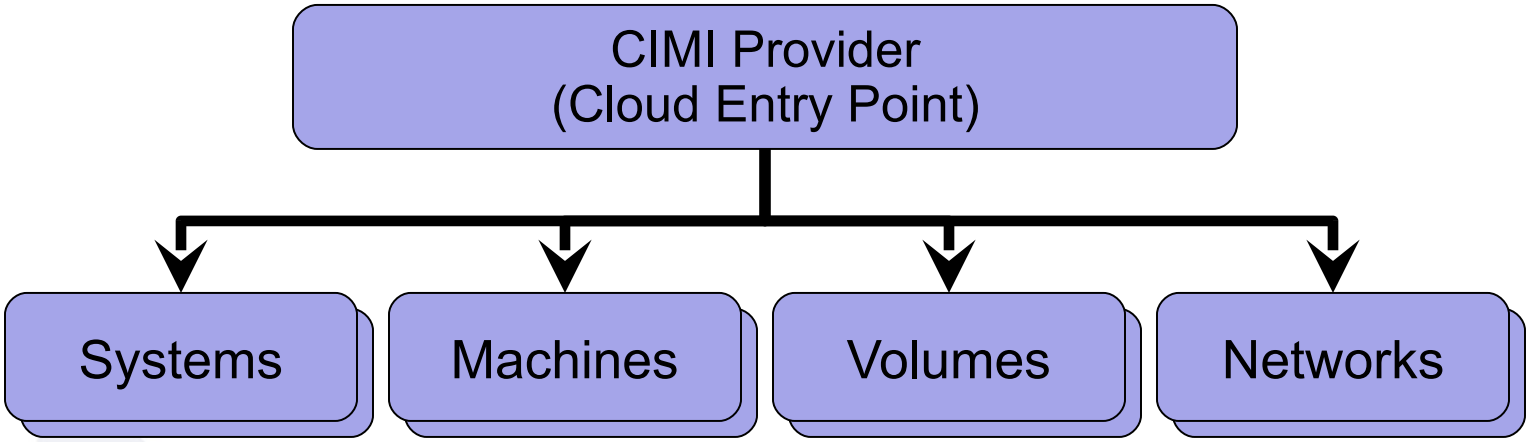
Main entry into the IaaS provider

All other data is discovered, iteratively:

Pointers to Machines, Volumes, Networks, etc...

Metadata describing capabilities and resources constraints

CIMI Model – Core Resources



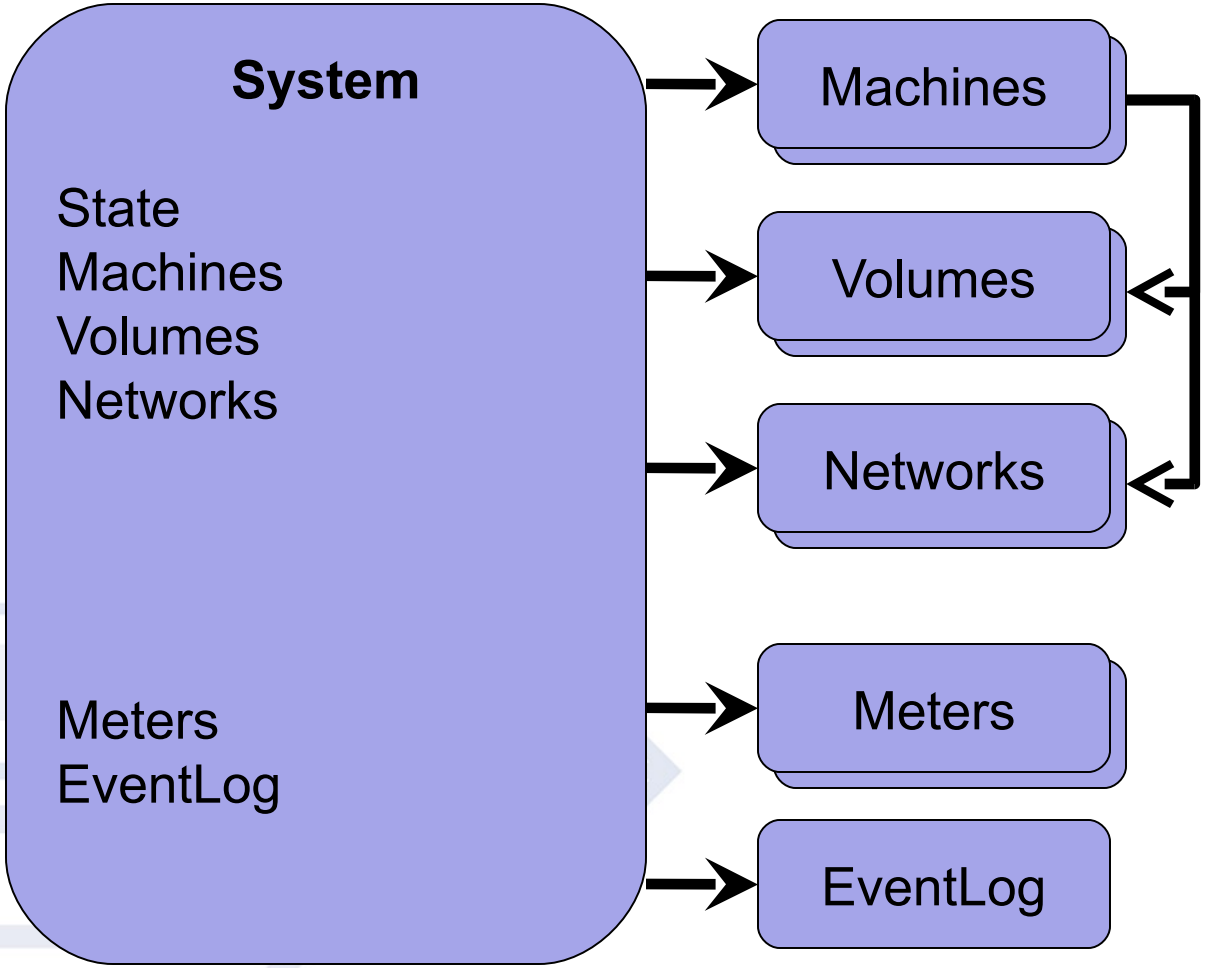
Grouping of resources meant to be managed as a single unit.

An instantiated compute resource that encapsulates both CPU and Memory.

A Volume represents storage at either the block or file-system level. Volumes can be attached to Machines.

A Network is a realized entity that represents an abstraction of a layer 2 broadcast domain.

CIMI Model - Systems

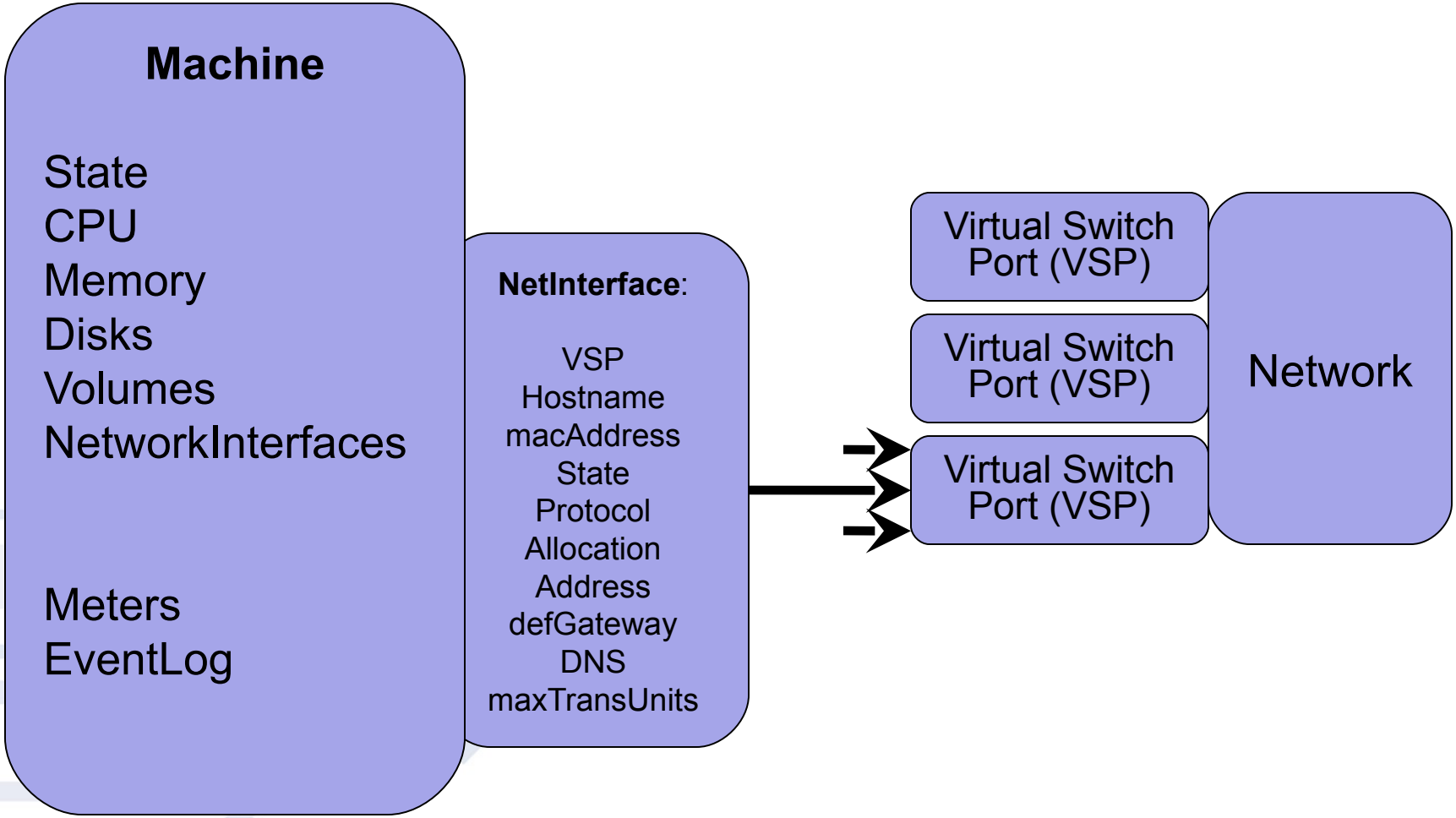


Grouping of resources meant to be managed as a single unit.

It is expected that System will be the import/export point for entire applications – e.g. an OVF file.

More on Meters and Events later...

CIMI Model – Machines & Networks



CIMI Model – Meters & Events & Jobs

Meters

- Represents an available Meter of some property associated to a given entity.
- Can take continuous or interval driven samples.
- Realized resources may have multiple Meters.

Events & Event Logs

- Event Logs are registries of Events.
  - Persistence duration is configurable.
  - Provides a summary (# of high, medium, low...) Events in the log
- Events are notifications of useful information from the Provider
  - Have time, type (error, warning...), severity (high, medium, low), contact info, ...

Jobs

- Represent a process/action performed by the Provider
- If supported, all operations (sync & async) generate Jobs

CIMI Model – Resource Metadata

Resources that describe the extensions and constraints of CIMI model.

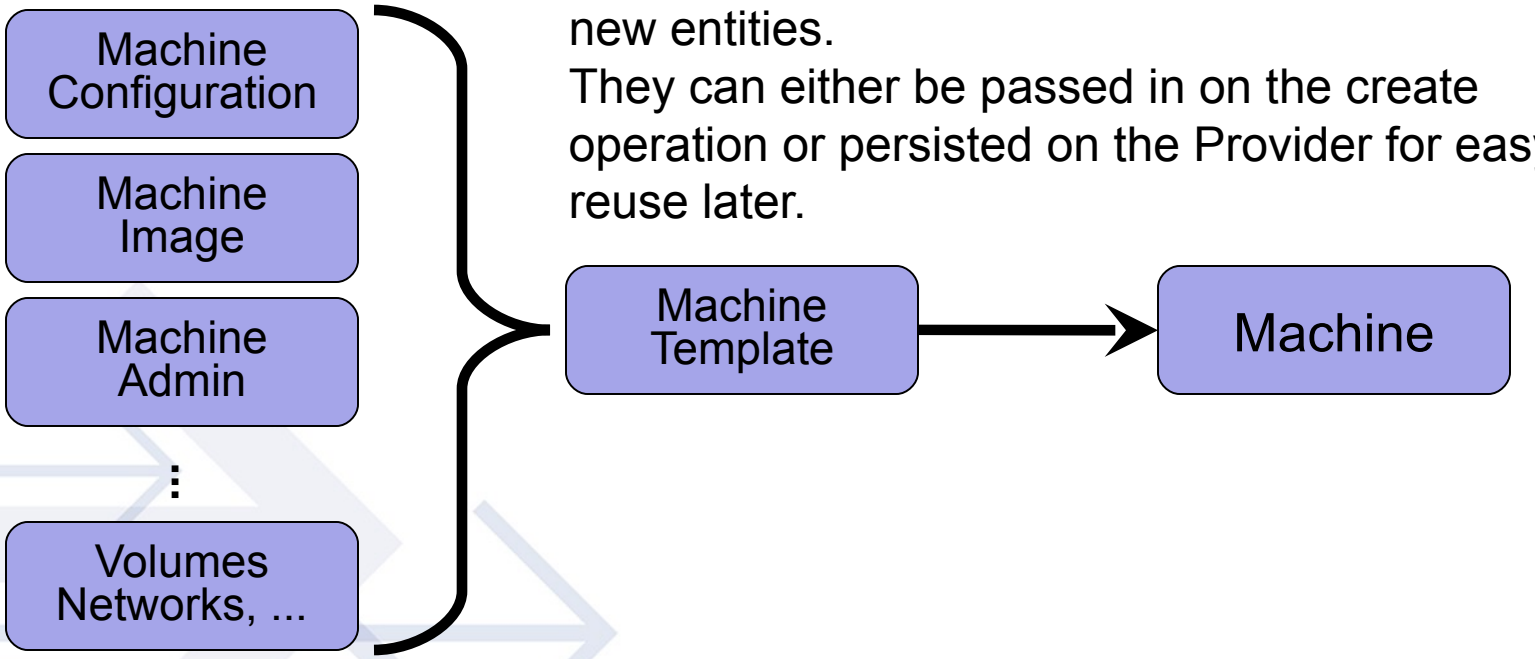
Part of the model itself so it's retrievable via the CloudEntryPoint.

Allows for machine discovery of model constraints.

Contains:

- List of extension attributes and operations
- Constraints on CIMI defined, and extension, attributes and extensions

CIMI Model – Resource Creation Pattern



Templates are the constructor data when creating new entities.
They can either be passed in on the create operation or persisted on the Provider for easy reuse later.

CIMI REST/HTTP-Based Protocol

Specification currently describes a REST/HTTP binding to the model.

Other bindings are possible.

This protocol binding follows REST principles and describes mapping of the HTTP protocol verbs to operations on the model.

Standard HTTP status codes are used to convey the results of the operations.

Serialization formats for the message body include JSON and XML

CIMI Primer

The CIMI Primer details specific scenarios and how to use the CIMI interface to accomplish them

- **Create a "Machine" – a running virtual machine image**
- Add a "Volume" to a Machine
- Defining and Using Machine Templates
- Create a new Machine from an existing Volume
- Defining and Using System Templates

- We'll take a look at the first scenario in detail – please read the Primer for more information on using the other scenarios

Creating a New Machine

This scenario will create a new Machine. The new Machine's configuration will be based on existing configurations and images offered by the provider. However, a new Credential resource (userid & password) will be created.

Retrieve the Cloud Entry Point (CEP)

The CEP will provide the links to the set of resources that are available in this Cloud. You retrieve the CEP to discover the URL to each collection:

```
GET / HTTP/1.1


HTTP/1.1 200 OK
Content-Type: application/json


{ "resourceURI": "http://schemas.dmtf.org/cimi/1/CloudEntryPoint",
  "id": "http://example.com/CEP",
  "baseURI": "http://example.com/",
  "resourceMetadata": { "href": "http://example.com/resourceMetadata" },
  "machines": { "href": "http://example.com/machines" },
  "machineConfigs": { "href": "http://example.com/machineConfigs" },
  "machineImages": { "href": "http://example.com/machineImages" },
  "credentials": { "href": "http://example.com/credentials" }
}
```

Creating a new Machine (1 of 3)

Retrieve the list of Machine Images

Before you can create a new Machine, first decide what kind of operating system and/or software you want to have pre-installed. The Machine Images collection is the set of Machine Images that this Cloud offers - note that some may be predefined by the Cloud while some may be user-created:

```
GET /machineImages HTTP/1.1


HTTP/1.1 200 OK
Content-Type: application/json


{ "resourceURI": "http://schemas.dmtf.org/cimi/1/MachineImageCollection",
  "id": "http://example.com/machineImages",
  "count": 3,
  "machineImages": [
    { "resourceURI": "http://schemas.dmtf.org/cimi/1/MachineImage",
      "id": "http://example.com/images/WinXP-SP2",
      "name": "WinXP SP2",
```

Creating a new Machine (2 of 3)


```
  "description": "Windows XP with Service Pack 2",
  "created": "2012-01-01T12:00:00Z",
  "updated": "2012-01-01T12:00:00Z",
  "imageLocation": { "href": "http://example.com/data/8934322" }
},

{
  "resourceURI": "http://schemas.dmtf.org/cimi/1/MachineImage",
  "id": "http://example.com/images/Win7",
  "name": "Windows 7",
  "description": "Windows 7",
  "created": "2012-01-01T12:00:00Z",
  "updated": "2012-01-01T12:00:00Z",
  "imageLocation": { "href": "http://example.com/data/8934344" }
},
```

Creating a new Machine (3 of 3)

```
    {
      "resourceURI": "http://schemas.dmtf.org/cimi/1/MachineImage",
      "id": "http://example.com/images/Linux-SUSE",
      "name": "Linux SUSE",
      "description": "Linux SUSE",
      "created": "2012-01-01T12:00:00Z",
      "updated": "2012-01-01T12:00:00Z",
      "imageLocation": { "href": "http://example.com/data/8934311" }
    }

  ]
}
```

Creating a New Machine

Choose a Machine Image

Next examine each Machine Image to find one that meets your needs. The first one is acceptable, so you will use it later.

It is worth noting that if you knew you wanted to use the first item in the list and only wanted to see that one resource in the previous query then the following could have been done instead:

GET /machineImages?$last=1 HTTP/1.1

HTTP/1.1 200 OK

Content-Type: application/json


```
{ "resourceURI": "http://schemas.dmtf.org/cimi/1/MachineImageCollection",
  "id": "http://example.com/machineImages",
  "count": 1,
  "machineImages": [
    { "resourceURI": "http://schemas.dmtf.org/cimi/1/MachineImage",
      "id": "http://example.com/images/WinXP-SP2",
      "name": "WinXP SP2",
      "description": "Windows XP with Service Pack 2",
      "created": "2012-01-01T12:00:00Z",
      "updated": "2012-01-01T12:00:00Z",
      "imageLocation": { "href": "http://example.com/data/8934322" }
    } ]
```

Creating a new Machine (1 of 3)

Retrieve the list of Machine Configurations

Next you decide what kind of virtual hardware you want to install your Machine Image onto. As with determining the kind of Machine Image you want, first ask for the list of available Machine Configurations:

GET /machineConfigs HTTP/1.1

HTTP/1.1 200 OK

Content-Type: application/json

{ "resourceURI":
"http://schemas.dmtf.org/cimi/1/MachineConfigurationCollection",
"id": "http://example.com/machineConfigs",
"count": 3,
"machineConfigurations": [
{ "resourceURI": "http://schemas.dmtf.org/cimi/1/MachineConfiguration",
"id": "http://example.com/configs/tiny",
"name": "tiny",
"description": "a teenie tiny one",

Creating a new Machine (2 of 3)

"created": "2012-01-01T12:00:00Z",
"updated": "2012-01-01T12:00:00Z",
"cpu": 1,
"memory": 4000000,
"disks" : [
{ "capacity": 50000000, "format": "ext4" }
]
},

{ "resourceURI": "http://schemas.dmtf.org/cimi/1/MachineConfiguration",
"id": "http://example.com/configs/small",
"name": "small",
"description": "a small sized one",
"created": "2012-01-01T12:00:00Z",
"updated": "2012-01-01T12:00:00Z",

Creating a new Machine (3 of 3)

```
"cpu": 1,
    "memory": 8000000,
    "disks" : [  { "capacity": 500000000, "format": "ext4" } ]
  },

  { "resourceURI": "http://schemas.dmtf.org/cimi/1/MachineConfiguration",
    "id": "http://example.com/configs/medium",
    "name": "medium",
    "description": "a medium one",
    "created": "2012-01-01T12:00:00Z",
    "updated": "2012-01-01T12:00:00Z",
    "cpu": 1,
    "memory": 16000000,
    "disks" : [
      { "capacity": 1000000000, "format": "ext4" },
      { "capacity": 1000000000, "format": "ext4" },
    ]}}
```

Creating a new Machine

Choose a Machine Configuration

Next examine the returned list and pick a Machine Configuration that suits your needs. The first one is acceptable, so you will use it later

Create a new Credential Resource

You want to use your own userid and password for this new Machine, so you need to create a new Credential resource. This process is done by using the POST operation, but first you need to retrieve the Credential collection so that you know where to POST a new Credential resource to:

GET /credentials HTTP/1.1

HTTP/1.1 200 OK

Content-Type: application/json


```
{ "resourceURI": "http://schemas.dmtf.org/cimi/1/CredentialCollection",
  "id": "http://example.com/credentials",
  "count": 0,
  "operations": [ { "rel": "add", "href": "http://example.com/credentials" }]
}
```

Creating a new Machine

Notice at this point there are no Credential resources in the environment. Before you can create a new Credential resource, you must first discover this Cloud provider's extension attributes for the Credential resource. By default the CIMI specification does not define how the initial user of a new Machine is specified; rather it is left open for each Cloud provider to determine how this information should be provided. Clients can discover this information by querying the Credential resource metadata resource. To examine this resource, first look through the ResourceMetadata collection for this Provider's description of the Credential's resource. Start by retrieving the ResourceMetadata collection from the URI referenced in the Cloud Entry Point:

GET /resourceMetadata HTTP/1.1

HTTP/1.1 200 OK

Content-Type: application/json

{ "resourceURI": "http://schemas.dmtf.org/cimi/1/ResourceMetadataCollection",
  "id": "http://example.com/resourceMetadata",
  "count": 1,
  "resourceMetadatas": [

Creating a new Machine

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/1/ResourceMetadata",
    "id": "http://example.com/resources/Credential",
    "typeURI": "http://schemas.dmtf.org/cimi/1/Credential",
    "name": "Credential",
    "attributes": [
      { "name": "userID", "namespace": "http://example.com",
        "type": "string", "required": "true" },
      { "name": "password", "namespace": "http://example.com",
        "type": "string", "required": "true" }
    ]
  }
 ]
}
```

Creating a new Machine

Now iterate over the list of resourceMetadata entries in the collection for the one whose "typeURI" is "http://schemas.dmtf.org/cimi/1/Credential". After you find it, you can now examine the extensions this Provider has added to the Credential resource and the above indicates that the Credential resource has been extended and must include two attributes called "userID" and "password". Both are of type "string".

Now create a new Credential resource by using the POST operation:

POST /credentials HTTP/1.1

Content-Type: application/json

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/1/CredentialCreate",
  "name": "Default",
  "description": "My Default User",
  "credentialTemplate": {
    "userID": "JoeSmith",
    "password": "letmein"
  }}
```

Creating a new Machine

HTTP/1.1 201 Created

Location: http://example.com/creds/12345

Note: While the "userID" and "password" attributes were discovered via the Credential ResourceMetadata, the "name" and "description" attributes are part of the common set of attributes available on all resources. In a future scenario you will see how the client knew that "userID" and "password" were the proper attribute names for this image type and Cloud provider.

Creating a new Machine

Create a new Machine

Retrieve the Machines collection so that you know where to POST a new Machine to:

GET /machines HTTP/1.1

HTTP/1.1 200 OK

Content-Type: application/json

{ "resourceURI": "http://schemas.dmtf.org/cimi/1/MachineCollection",
"id": "http://example.com/machines",
"count": 0,
"operations": [{ "rel": "add", "href": "http://example.com/machines" }]
}

Creating a new Machine

Now create a new one:

POST /machines HTTP/1.1

Content-Type: application/json

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/1/MachineCreate",
  "name": "myMachine1",
  "description": "My very first machine",
  "machineTemplate": {
    "machineConfig": { "href": " http://example.com/configs/tiny" },
    "machineImage": { "href": " http://example.com/images/WinXP-SP2" },
    "credential": { "href": "http://example.com/creds/12345" }
  }
}
```

Creating a new Machine

HTTP/1.1 201 Created

Location: http://example.com/machines/843752

Note that the Provider could have chosen to return the representation of the new Machine in the HTTP response, thus making the next step redundant.

Query new Machine

Retrieve the Machine to get the full representation of the new Machine:

GET /machines/843752 HTTP/1.1

Creating a new Machine

HTTP/1.1 200 OK
Content-Type: application/json

{ "resourceURI": "http://schemas.dmtf.org/cimi/1/Machine",
  "id": "http://example.com/machines/843752",
  "name": "myMachine1",
  "description": "My very first machine",
  "created": "2012-08-15T12:15:00Z",
  "updated": "2012-08-15T12:15:00Z",
  "state": "STOPPED",
  "cpu": 1,
  "memory": 4000000,
  "disks" : { "href": "http://example.com/machines/843752/disks",

Creating a new Machine

"networkInterfaces": { "href": "http://example.com/machines/843752/NIs",
  "operations": [
    { "rel": "edit", "href": "http://example.com/machines/843752" },
    { "rel": "delete", "href": "http://example.com/machines/843752" },
    { "rel": "http://schemas.dmtf.org/cimi/1/action/start",
      "href": "http://example.com/machines/843752" }
 ]
}

Notice the "state"' attribute on the Machine is "STOPPED", which is the initial state of a new machine.

Creating a new Machine

Start a Machine

The presence of the "start" operation in the "operations" array of the Machine representation indicates not only which URI to POST the "start" operation to, but that you are able to do it at this time.

POST /machines/843752 HTTP/1.1

Content-Type: application/json

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/1/Action",
  "action": "http://schemas.dmtf.org/cimi/1/action/start"
}
```

HTTP/1.1 204 No Content

Creating a new Machine

Query a Machine to verify it is started

Query the Machine again to verify that it is started:

GET /machines/843752 HTTP/1.1

HTTP/1.1 200 OK

Content-Type: application/json

{ "resourceURI": "http://schemas.dmtf.org/cimi/1/Machine",
  "id": "http://example.com/machines/843752",
  "name": "myMachine1",
  "description": "My very first machine",
  "created": "2012-08-15T12:15:00Z",
  "updated": "2012-08-15T12:15:00Z",
  "state": "STARTED",

Creating a new Machine


```
"cpu": 1,
  "memory": 4000000,
  "disks" : { "href": "http://example.com/machines/843752/disks",
  "networkInterfaces": { "href": "http://example.com/machines/843752/NIs",
  "operations": [
    { "rel": "edit", "href": "http://example.com/machines/843752" },
    { "rel": "delete", "href": "http://example.com/machines/843752" },
    { "rel": "http://schemas.dmtf.org/cimi/1/action/stop",
      "href": "http://example.com/machines/843752" }
  ]
}
```


Notice the "state" attribute on the Machine is "STARTED" and that the "operations" array no longer indicates that the "start" operation is available; rather the "stop" operation is available now instead.

Creating a new Machine

Stop a Machine

Using the "stop" operation's URL, you can now ask for the Machine to be stopped:

POST /machines/843752 HTTP/1.1

Content-Type: application/json

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/1/Action",
  "action": "http://schemas.dmtf.org/cimi/1/action/stop"
}
```

HTTP/1.1 204 No Content

Creating a new Machine

Update a Machine's attributes

Using PUT operation on the "edit" operation's URL, you can update some of the attribute of the Machine, for example the "name" and "description":

PUT /machines/843752?$select=name,description HTTP/1.1

Content-Type: application/json

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/1/Machine",
  "name" : "Cool Demo #1"
}
```

HTTP/1.1 200 OK

```
{ "name" : "Cool Demo #1" }
```

Notice that URL of the "edit" operation has been modified to indicate which attributes are being updated; only those attributes will be touched. Because the URL includes the "description" attribute but the HTTP request body does not, that attribute is erased.

Questions?

Thank you!

Dmtf.org/cloud