



# Cloud Infrastructure Management Interface (CIMI) Primer

**Information for Work-in-Progress version:**

**IMPORTANT:** This specification is not a standard. It does not necessarily reflect the views of the DMTF or all of its members. Because this document is a Work in Progress, this specification may still change, perhaps profoundly. This document is available for public review and comment until the stated expiration date.

**It expires on: 2012-03-15**

**Provide any comments through the DMTF Feedback Portal:**

<http://www.dmtf.org/standards/feedback>

**Version: 0.0.16**

**Status: Work In Progress**

**Publication Date: 2012-01-04**

**Document Number: DSP2027**

**Copyright Notice**

Copyright © 2011 Distributed Management Task Force, Inc. (DMTF). All rights reserved.

DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability. Members and non-members may reproduce DMTF specifications and documents, provided that correct attribution is given. As DMTF specifications may be revised from time to time, the particular version and release date should always be noted. Implementation of certain elements of this standard or proposed standard may be subject to third party patent rights, including provisional patent rights (herein "patent rights"). DMTF makes no representations to users of the standard as to the existence of such rights, and is not responsible to recognize, disclose, or identify any or all such third party patent right, owners or claimants, nor for any incomplete or inaccurate identification or disclosure of such rights, owners or claimants. DMTF shall have no liability to any party, in any manner or circumstance, under any legal theory whatsoever, for failure to recognize, disclose, or identify any such third party patent rights, or for such party's reliance on the standard or incorporation thereof in its product, protocols or testing procedures. DMTF shall have no liability to any party implementing such standard, whether such implementation is foreseeable or not, nor to any patent owner or claimant, and shall have no liability or responsibility for costs or losses incurred if a standard is withdrawn or modified after publication, and shall be indemnified and held harmless by any party implementing the standard from any and all claims of infringement by a patent owner for such implementations.

For information about patents held by third-parties which have notified the DMTF that, in their opinion, such patent may relate to or impact implementations of DMTF standards, visit <http://www.dmtf.org/about/policies/disclosures.php>.

**Abstract**

This document contains scenarios that describe common uses of the CIMI protocol.

For the sake of simplicity, in each of the following scenarios the Cloud Provider only supports the minimum needed to demonstrate the features highlighted by each scenario.

**Contents**

**Abstract..... 3**

**Contents ..... 4**

**Scenario 1 : Create a New Machine ..... 5**

    Step 1 : Cloud Entry Point (CEP) URL is provided: ..... 5

    Step 2 : Retrieve the CEP..... 5

    Step 3 : Retrieve the list of Machine Images ..... 5

    Step 4 : Retrieving the list of Machine Configurations ..... 6

    Step 5 : Create a new MachineAdmin ..... 7

    Step 6 : Create a new Machine ..... 8

    Step 7 : Querying new Machine..... 9

    Step 8 : Stop a Machine ..... 9

    Step 9 : Start a Machine ..... 10

    Step 10 : Update a Machine's properties..... 10

**Scenario 2 : Adding a New Volume to a Machine..... 12**

    Step 1 : Cloud Entry Point (CEP) and Machine URLs are provided..... 12

    Step 2 : Retrieve the CEP..... 12

    Step 3 : Get the list of VolumeConfigurations to determine which to use ..... 12

    Step 4 : Create a new Volume..... 13

    Step 5 : Attach the new Volume to a Machine ..... 13

    Step 6 : Querying the Machine to verify the update ..... 14

    Step 7 : Retrieve the Volume Information..... 15

**Scenario 3 : Defining and Using Machine Templates..... 16**

    Step 1 : Cloud Entry Point (CEP) is provided ..... 16

    Step 2 : Retrieve the CEP..... 16

    Step 3 : Create a new Machine Template ..... 16

    Step 4 : Create a new Machine using a Machine Template..... 17

**Change History: ..... 18**

## **Scenario 1 : Create 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 Machine Admin entity (userid & password) will be created.

### **Step 1 : Cloud Entry Point (CEP) URL is provided:**

In order to interact with the provider, we must first be provided a URL to the main entry point (CEP). How we obtained this URL is out of scope of the specification:

```
http://example.com
```

### **Step 2 : Retrieve the CEP**

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

```
GET / HTTP/1.1
```

```
HTTP/1.1 200 OK
Content-Type: application/CIMI-CloudEntryPoint+json
X-CIMI-Specification-Version: 1.0

{ "self": "http://example.com",
  "machineConfigs": { "href": "http://example.com/machineConfigs" },
  "machineImages": { "href": "http://example.com/machineImages" },
  "machineAdmins": { "href": "http://example.com/machineAdmins" },
  "machines": { "href": "http://example.com/machines" },
  "entityMetadata": [
    { "rel": "http://www.dmtf.org/cimi/MachineAdmin",
      "href": "http://example.com/entities/MachineAdmin" }
  ]
}
```

### **Step 3 : Retrieve the list of Machine Images**

Before we can create a new Machine we must first decide what kind of operating system and/or software we want to have pre-installed. The Machine Images collection is the set of predefined Machine Images that this Cloud offers:

```
GET /machineImages HTTP/1.1
```

```
HTTP/1.1 200 OK
Content-Type: application/CIMI-MachineImageCollection+json
X-CIMI-Specification-Version: 1.0

{ "self": "http://example.com/machineImages",
  "machineImages": [
    { "href": "http://example.com/images/WinXP-SP2" },
    { "href": "http://example.com/images/Win7" },
    { "href": "http://example.com/images/Linux-SUSE" }
  ]
}
```

### Step 3.1 : Choosing a Machine Image

---

Next we examine each Machine Image to find one that meets our needs. Normally this would involve iterating over the list of Machine Image URIs returned in the previous step and retrieving/examining each Machine Image resource - for brevity sake we'll only the first one:

```
GET /images/WinXP-SP2 HTTP/1.1
```

```
HTTP/1.1 200 OK
Content-Type: application/CIMI-MachineImage+json
X-CIMI-Specification-Version: 1.0

{ "self": "http://example.com/images/WinXP-SP2",
  "name": "WinXP SP2",
  "description": "Windows XP with Service Pack 2",
  "created": "2011/01/01 12:00:00pm",
  "imageLocation": { "href": "http://example.com/data/8934322" }
}
```

We like this one so we'll use it later.

### Step 4 : Retrieving the list of Machine Configurations

---

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

```
GET /machineConfigs HTTP/1.1
```

```
HTTP/1.1 200 OK
Content-Type: application/CIMI-MachineConfigurationCollection+json
X-CIMI-Specification-Version: 1.0

{ "self": "http://example.com/machineConfigs",
  "machineConfigurations": [
    { "href": "http://example.com/configs/small" },
    { "href": "http://example.com/configs/medium" },
    { "href": "http://example.com/configs/large" }
  ]
}
```

#### Step 4.1 : Choosing a Machine Configuration

---

Next we examine the returned list and pick a Machine Configuration that suits our needs. Normally this would involve iterating over the list of Machine Configuration URIs returned in the previous step and retrieving/examining each Machine Configuration resource - for brevity sake we'll only the first one:

```
GET /configs/small HTTP/1.1
```

```
HTTP/1.1 200 OK
Content-Type: application/CIMI-MachineConfiguration+json
X-CIMI-Specification-Version: 1.0

{ "self": "http://example.com/configs/small",
  "name": "small",
  "description": "a teenie tiny one",
  "created": "2010/01/01 12:00:00pm",
```

```

"cpu": "1",
"memory": { "quantity": 4, "units": "gibibyte" },
"disks" : [
  { "capacity": { "quantity": 50, "units": "gigabyte" } }
]
}

```

We like this one so we'll use it later.

### Step 5 : Create a new MachineAdmin

We want to use our own userid and password for this new Machine so we need to create a new MachineAdmin entity. First we need to retrieve the MachineAdmin collection so we know where to POST a new Admin resource to:

```
GET /machineAdmins HTTP/1.1
```

```

HTTP/1.1 200 OK
Content-Type: application/CIMI-MachineAdminCollection+json
X-CIMI-Specification-Version: 1.0

{ "self": "http://example.com/machineAdmins",
  "machineAdmins": [ ],
  "operations": [
    { "rel": "add", "href": "http://example.com/machineAdmins" }
  ]
}

```

Before we can create a new MachineAdmin entity we must first discover this Cloud provider's extension attributes for the MachineAdmin entity. 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 MachineAdmin entity metadata resource. The URL to this resource was provided in the CEP's representation:

```
GET /entities/MachineAdmin HTTP/1.1
```

```

HTTP/1.1 200 OK
Content-Type: application/CIMI-EntityMetadata+json
X-CIMI-Specification-Version: 1.0

{ "self": "http://example.com/entities/MachineAdmin",
  "typeURI": "http://www.dmtf.org/cimi/MachineAdmin",
  "name": "MachineAdmin",
  "attributes": [
    { "name": "userID", "namespace": "http://example.com",
      "type": "string", "required": "true" },
    { "name": "password", "namespace": "http://example.com",
      "type": "string", "required": "true" }
  ]
}

```

The above indicates that the MachineAdmin entity has been extended and must include two attributes called "userID" and "password" - both are of type "string".

Now create a new one:

```
POST /machineAdmins HTTP/1.1
Content-Type: application/CIMI-MachineAdminCreate+json
X-CIMI-Specification-Version: 1.0

{
  "name": "Default",
  "description": "My Default User",
  "machineAdminTemplate": {
    "userID": "JoeSmith",
    "password": "letmein"
  }
}
```

```
HTTP/1.1 201 Created
Location: http://example.com/admins/12345
X-CIMI-Specification-Version: 1.0
```

Note: In a future scenario we will discuss how the client knew that "userID" and "password" were the proper attribute names for this image type and Cloud provider.

### Step 6 : Create a new Machine

---

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

```
GET /machines HTTP/1.1
```

```
HTTP/1.1 200 OK
Content-Type: application/CIMI-MachineCollection+json
X-CIMI-Specification-Version: 1.0

{
  "self": "http://example.com/machines",
  "machines": [],
  "operations": [
    { "rel": "add", "href": "http://example.com/machines" }
  ]
}
```

Now create a new one:

```
POST /machines HTTP/1.1
Content-Type: application/CIMI-MachineCreate+json
X-CIMI-Specification-Version: 1.0

{
  "name": "myMachine1",
  "description": "My very first machine",
  "machineTemplate": {
    "machineConfig": { "href": " http://example.com/configs/small" },
    "machineImage": { "href": " http://example.com/images/WinXP-SP2" },
    "machineAdmin": { "href": "http://example.com/admins/12345" }
  }
}
```

```
HTTP/1.1 201 Created
Location: http://example.com/machines/843752
X-CIMI-Specification-Version: 1.0
```

## Step 7 : Querying new Machine

---

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

```
GET /machines/843752 HTTP/1.1
```

```
HTTP/1.1 200 OK
Content-Type: application/CIMI-Machine+json
X-CIMI-Specification-Version: 1.0

{ "self": "http://example.com/machines/843752",
  "name": "myMachine1",
  "description": "My very first machine",
  "created": "2011/08/15 12:15:00pm",
  "statzus": "STARTED",
  "cpu": "1",
  "memory": { "quantity": 4, "units": "gibibyte" },
  "disks" : [
    { "capacity": { "quantity": 50, "units": "gigabyte" } }
  ],
  "networkInterfaces": [
    { "vsp": { "href": "..."},
      "hostname": "exampleHost123",
      "macAddress": "...",
      "state": "Active",
      "protocol": "IPv4",
      "allocation": "DHCP",
      "address": "23.12.42.65",
      "defaultGateway": "23.12.42.0",
      "dns": "23.12.1.1",
      "maxTransmissionUnit": 1024 }
    ]
  "operations": [
    { "rel": "edit", "href": "http://example.com/machines/843752" },
    { "rel": "delete", "href": "http://example.com/machines/843752" },
    { "rel": "http://www.dmtf.org/cimi/action/stop",
      "href": "http://example.com/machines/843752" }
    ]
}
```

## Step 8 : Stop a Machine

---

The "operations" array in the Machine representation indicates not only which URI to POST the "stop" command to, but that we are able to do it at this time.

```
POST /machines/843752 HTTP/1.1
```

```
Content-Type: application/CIMI-Action+json
```

```
{ "action": "http://www.dmtf.org/cimi/action/stop" }
```

```
HTTP/1.1 204 No Content
```

```
X-CIMI-Specification-Version: 1.0
```

### Step 8.1 : Query a Machine to verify it is stopped

---

Query the Machine again to verify that it is stopped:

```
GET /machines/843752 HTTP/1.1
```

```

HTTP/1.1 200 OK
Content-Type: application/CIMI-Machine+json
X-CIMI-Specification-Version: 1.0

{ "self": "http://example.com/machines/843752",
  "name": "myMachine1",
  "description": "My very first machine",
  "created": "2011/08/15 12:15:00pm",
  "state": "STOPPED",
  "cpu": "1",
  "memory": { "quantity": 4, "units": "gibibyte" },
  "disks" : [
    { "capacity": { "quantity": 50, "units": "gigabyte" } }
  ],
  "networkInterfaces": [
    { "vsp": {"href": "..."},
      "hostname": "exampleHost123",
      "macAddress": "...",
      "state": "Active",
      "protocol": "IPv4",
      "allocation": "DHCP",
      "address": "23.12.42.65",
      "defaultGateway": "23.12.42.0",
      "dns": "23.12.1.1",
      "maxTransmissionUnit": 1024 }
    ]
  "operations": [
    { "rel": "edit", "href": "http://example.com/machines/843752" },
    { "rel": "delete", "href": "http://example.com/machines/843752" },
    { "rel": "http://www.dmtf.org/cimi/action/start",
      "href": "http://example.com/machines/843752" }
    ]
  }

```

Notice the 'state' property on the Machine is "STOPPED" and that the "operations" array no longer indicated that the "stop" operation is available; rather the "start" operation is available now instead.

### Step 9 : Start a Machine

Using the "start" operation's URL, we can now ask for the Machine to be started:

```

POST /machines/843752 HTTP/1.1
Content-Type: application/CIMI-Action+json

{ "action": "http://www.dmtf.org/cimi/action/start" }

```

```

HTTP/1.1 204 No Content
X-CIMI-Specification-Version: 1.0

```

### Step 10 : Update a Machine's properties

Using the "edit" operation's URL, we can update some of the properties of the Machine - in this case the "name" and "description":

```

PUT /machines/843752?CIMISelect=name,description HTTP/1.1
Content-Type: application/CIMI-Machine+json

```

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

```
HTTP/1.1 204 No Content  
X-CIMI-Specification-Version: 1.0
```

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

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

## **Scenario 2 : Adding a New Volume to a Machine**

This scenario will create a new Volume and attach it to an existing Machine. This simple example assumes that the user knows that the machine currently has no volumes attached. If additional volumes may be present, additional steps are necessary.

### **Step 1 : Cloud Entry Point (CEP) and Machine URLs are provided**

---

CEP:

```
http://example.com
```

Machine:

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

### **Step 2 : Retrieve the CEP**

---

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

```
GET / HTTP/1.1 HTTP/1.1
```

```
HTTP/1.1 200 OK
Content-Type: application/CIMI-CloudEntryPoint+json
X-CIMI-Specification-Version: 1.0

{ "self": "http://example.com",
  "machineConfigs": { "href": "http://example.com/machineConfigs" },
  "machineImages": { "href": "http://example.com/machineImages" },
  "machineAdmins": { "href": "http://example.com/machineAdmins" },
  "machines": { "href": "http://example.com/machines" },
  "volumeConfigs": { "href": "http://example.com/vConfigs" },
  "volumes": { "href": "http://example.com/volumes" }
}
```

### **Step 3 : Get the list of VolumeConfigurations to determine which to use**

---

When creating a new Volume we need to decide what kind of Volume to create - e.g. its size, format, etc. The Volume Configurations collection is the set of predefined Volume Configurations that this Cloud offers:

```
GET /vConfigs HTTP/1.1
```

```
HTTP/1.1 200 OK
Content-Type: application/CIMI-VolumeConfigurationCollection+json
X-CIMI-Specification-Version: 1.0

{ "self": "http://example.com/vConfigs",
  "volumeConfigurations": [
    { "href": "http://example.com/vConfigs/small" },
    { "href": "http://example.com/vConfigs/medium" },
    { "href": "http://example.com/vConfigs/large" }
  ]
}
```

#### **Step 3.1 : Choosing a Volume Configuration**

---

Next we examine each Volume Configuration to find the one that meets our needs. Normally this would involve iterating over the list of Volume Configurations URIs returned in the previous

step and retrieving/examining each Volume Configuration resource - for brevity sake we'll only the first one:

```
GET /vConfigs/small HTTP/1.1
```

```
HTTP/1.1 200 OK
Content-Type: application/CIMI-VolumeConfigurationjson
X-CIMI-Specification-Version: 1.0

{ "self": "http://example.com/vConfigs/small",
  "name": "Small",
  "description": "A pretty small one",
  "format": "NTFS",
  "capacity": { "quantity": 60, "units": "gigabyte" },
  "supportsSnapshots": false,
  "guestInterface": "NFS",
}
```

We like this one so we'll use it later.

#### Step 4 : Create a new Volume

---

Retrieve the Volumes collection so we know where to POST a new Volume to:

```
GET /volumes HTTP/1.1
```

```
HTTP/1.1 200 OK
Content-Type: application/CIMI-VolumeCollection+json
X-CIMI-Specification-Version: 1.0

{ "self": "http://example.com/volumes",
  "operations": [
    { "rel": "add", "href": "http://example.com/volumes" }
  ]
}
```

Now create a new one:

```
POST /volumes HTTP/1.1
Content-Type: application/CIMI-VolumeCreate+json
X-CIMI-Specification-Version: 1.0

{ "name": "myVolume1",
  "description": "My first new volume",
  "volumeTemplate": {
    "volumeConfig": { "href": "http://example.com/vConfigs/small" }
  }
}
```

```
HTTP/1.1 201 Created
Location: http://example.com/volumes/35782
X-CIMI-Specification-Version: 1.0
```

#### Step 5 : Attach the new Volume to a Machine

---

Update the "volumes" attribute of the Machine so it points to this new Volume:

```
PUT /machines/843752?CIMISelect=volumes HTTP/1.1
Content-Type: application/CIMI-Machine+json
```

```
{ "volumes" : [
  { "volume": { "href": "http://example.com/volumes/35782" },
    "attachmentPoint": "V" }
]
```

```
HTTP/1.1 204 No Content
X-CIMI-Specification-Version: 1.0
```

```
{ "volumes" : [
  { "volume": { "href": "http://example.com/volumes/35782" },
    "attachmentPoint": "V" }
]
```

In a more complex example where additional volumes may be present, the simple PUT of the new volume in this step would remove existing volumes and replace them with the single new volume. When this possibility exists, the user should retrieve the list of volumes and PUT the entire list with the new volume appended.

### Step 6 : Querying the Machine to verify the update

Retrieve the Machine to get its full representation to verify the update was successful:

```
GET /machines/843752 HTTP/1.1
```

```
HTTP/1.1 200 OK
Content-Type: application/CIMI-Machine+json
X-CIMI-Specification-Version: 1.0
```

```
{ "self": "http://example.com/machines/843752",
  "name": "myMachine1",
  "description": "My very first machine",
  "created": "2011/08/15 12:15:00pm",
  "state": "STARTED",
  "cpu": "1",
  "memory": { "quantity": 4, "units": "gibibyte" },
  "disks" : [
    { "capacity": { "quantity": 50, "units": "gigabyte" } }
  ],
  "volumes" : [
    { "volume": { "href": "http://example.com/volumes/35782" },
      "attachmentPoint": "V" }
  ],
  "networkInterfaces": [
    { "vsp": { "href": "..."},
      "hostname": "exampleHost123",
      "macAddress": "...",
      "state": "Active",
      "protocol": "IPv4",
      "allocation": "DHCP",
      "address": "23.12.42.65",
      "defaultGateway": "23.12.42.0",
      "dns": "23.12.1.1",
      "maxTransmissionUnit": 1024 }
  ]
  "operations": [
```

```
{ "rel": "edit", "href": "http://example.com/machines/843752" },  
  { "rel": "delete", "href": "http://example.com/machines/843752" },  
  { "rel": "http://www.dmtf.org/cimi/action/stop",  
    "href": "http://example.com/machines/843752" }  
]  
}
```

### Step 7 : Retrieve the Volume Information

---

To verify that the Volume we created and attached to the Machine is what we wanted, we follow the reference returned from the previous step:

```
GET /volumes/35782 HTTP/1.1
```

```
HTTP/1.1 200 OK  
Content-Type: application/CIMI-Volume+json  
X-CIMI-Specification-Version: 1.0  
  
{ "self": "http://example.com/volumes/35782",  
  "name": "myVolume1",  
  "description": "My first new volume",  
  "capacity": { "quantity": 60, "units": "Gigabyte" },  
  "supportsSnapshots": false,  
  "guestInterface": "NFS",  
  "operations": [  
    { "rel": "edit", "href": "http://example.com/volumes/35782" },  
    { "rel": "delete", "href": "http://example.com/volumes/35782" }  
  ]  
}
```

### **Scenario 3 : Defining and Using Machine Templates**

This scenario will create a new Machine Template that will then be used to create a new Machine. Machine Templates are convenience entities that allow for well-defined descriptions (configuration, image, etc) of a Machine to be persisted such that it can be reused later. This is particularly useful when the user of the new Machine may not be technically savvy enough to know all of the details necessary to create the Machine. Another common use is to have Machine Templates be created for demos, or complex configurations, where a particular Machine Image must be used on a particular Machine Configuration. Machine Templates allow this information to be persisted and easily reused.

For convenience, we will reuse the configuration information already obtained in the previous scenarios.

#### **Step 1 : Cloud Entry Point (CEP) is provided**

---

CEP:

```
http://example.com
```

#### **Step 2 : Retrieve the CEP**

---

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

```
GET / HTTP/1.1
```

```
HTTP/1.1 200 OK
Content-Type: application/CIMI-CloudEntryPoint+json
X-CIMI-Specification-Version: 1.0

{
  "self": "http://example.com",
  "machineTemplates": { "href": "http://example.com/machineTemplates" },
  "machineConfigs": { "href": "http://example.com/machineConfigs" },
  "machineImages": { "href": "http://example.com/machineImages" },
  "machineAdmins": { "href": "http://example.com/machineAdmins" },
  "machines": { "href": "http://example.com/machines" }
}
```

#### **Step 3 : Create a new Machine Template**

---

From the previous scenarios we already have the MachineConfiguration, MachineImage and MachineAdmin resources that we will reuse for this MachineTemplate:

MachineImage:

```
http://example.com/images/WinXP-SP2
```

MachineConfiguration:

```
http://example.com/configs/small
```

MachineAdmin:

```
http://example.com/admins/12345
```

Before we can create the new MachineTemplate we need to first determine the URL to which the POST is sent. This is obtained from the MachineTemplate collection URL that was returned as part of the CEP:

```
GET /machineTemplates HTTP/1.1
```

```
HTTP/1.1 200 OK
```

```
Content-Type: application/CIMI-MachineTemplateCollection+json
```

```
X-CIMI-Specification-Version: 1.0
```

```
{ "self": "http://example.com/machineTemplates",  
  "operations": [  
    { "rel": "add", "href": "http://example.com/machineTemplates" }  
  ]  
}
```

Now we create the new MachineTemplate entity:

```
POST /machineTemplates HTTP/1.1
```

```
Content-Type: application/CIMI-MachineTemplate+json
```

```
X-CIMI-Specification-Version: 1.0
```

```
{ "name": "Demo1",  
  "description": "My first demo",  
  "machineConfig": { "href": "http://example.com/configs/small" },  
  "machineImage": { "href": "http://example.com/images/WinXP-SP2" },  
  "machineAdmin": { "href": "http://example.com/admins/12345" }  
}
```

```
HTTP/1.1 201 Created
```

```
Location: http://example.com/machineTemplates/82754
```

```
X-CIMI-Specification-Version: 1.0
```

#### Step 4 : Create a new Machine using a Machine Template

---

Now create a new Machine using this Machine Template:

```
POST /machines HTTP/1.1
```

```
Content-Type: application/CIMI-MachineCreate+json
```

```
X-CIMI-Specification-Version: 1.0
```

```
{ "name": "myMachine2",  
  "description": "My second machine",  
  "machineTemplate": { "href": "http://example.com/machineTemplates/82754" }  
}
```

```
HTTP/1.1 201 Created
```

```
Location: http://example.com/machines/843799
```

```
X-CIMI-Specification-Version: 1.0
```

**Change History:**

| Version | Date     | Who  | What   |
|---------|----------|------|--|
| 0.0.1   | 08/26/11 | Doug | Initial Draft  |
| 0.0.2   | 08/28/11 | Doug | Added creating/attaching a Volume scenario. Added the notion of checking the EntityMetadata for MachineAdmin in scenario #1 since the client needs to know that userid/pwd are required. Added change history. |
| 0.0.3   | 08/29/11 | Doug | Added creating/using MachineTemplates. Renamed the doc to be a Primer.   |
| 0.0.4   | 08/30/11 | Doug | Added more descriptive text to indicate that in a real world scenario people would normally loop over the entire list of resources returned looking for the one they want.                                     |
| 0.0.5   | 08/31/11 | Doug | Added boilerplate DMTF info.   |
| 0.0.6   | 09/09/11 | Doug | Added a step to scenario to update some properties.  |
| 0.0.8   | 09/21/11 | Doug | Make 'stop' action URIs consistent to resolve issue <a href="#">1364</a> .   |
| 0.0.9   | 10/04/11 | Doug | s/gigabyte/gibibyte/ in memory to resolve issue <a href="#">1362</a> .   |
| 0.0.10  | 10/04/11 | Doug | Alignment with issues <a href="#">1241</a> , <a href="#">1242</a> , <a href="#">1351</a> , <a href="#">1220</a> resolved in v42 of the main spec.  |
| 0.0.11  | 10/05/11 | Doug | Alignment with issue <a href="#">1369</a> .  |
| 0.0.12  | 10/06/11 | Doug | s/status/state/ to align with issue <a href="#">1095</a> . Make sure PUT responses have response bodies to align with issue <a href="#">1118</a> .   |
| 0.0.13  | 10/26/11 | Doug | Added some abstract text to resolve issue <a href="#">1422</a> .   |
| 0.0.14  | 12/06/11 | Doug | Added clarifying text around adding volumes to resolve issue <a href="#">1441</a> .  |
| 0.0.15  | 12/07/11 | Doug | Removed "protocol" from Machine and Volume entities to resolve issue <a href="#">1247</a> .  |
| 0.0.16  | 12/08/11 | Doug | Added the use of MachineAdminTemplate to resolve issue <a href="#">1368</a> .  |