Copyright notice

Copyright © 2014-2016 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patent owner or claimant, and shall have no liability or responsibility for costs or losses incurred if a standard is withdrawn or modified after publication, and shall be indemnified and held harmless by any party implementing the standard from any and all claims of infringement by a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.

This document’s normative language is English. Translation into other languages is permitted.
CONTENTS

1 Common scenarios of consumer-provider interactions .............................................. 11
  1.1 Creating a new Machine ...................................................................................... 11
    1.1.1 Retrieve the CEP ............................................................................................ 11
    1.1.2 Retrieve the list of Machine Images ............................................................... 11
    1.1.3 Choose a Machine Image .............................................................................. 12
    1.1.4 Retrieve the list of Machine Configurations ............................................... 12
    1.1.5 Choose a Machine Configuration ................................................................ 13
    1.1.6 Create a new Credential Resource ............................................................. 14
    1.1.7 Create a new Machine ................................................................................... 15
    1.1.8 Query new Machine ...................................................................................... 16
    1.1.9 Start a Machine ............................................................................................ 16
    1.1.10 Query a Machine ......................................................................................... 17
    1.1.11 Stop a Machine .......................................................................................... 17
    1.1.12 Update a Machine’s attributes ................................................................. 17
  1.2 Adding a new Volume to a Machine ................................................................. 18
    1.2.1 Obtain the Machine URL .............................................................................. 18
    1.2.2 Retrieve the u .............................................................................................. 18
    1.2.3 Get the list of VolumeConfigurations .......................................................... 18
    1.2.4 Choose a Volume Configuration ................................................................ 19
    1.2.5 Create a new Volume ................................................................................... 19
    1.2.6 Retrieve the Volume information ................................................................. 20
    1.2.7 Retrieve the Machine's Volume collection .................................................. 20
    1.2.8 Connect the new Volume to a Machine ...................................................... 22
    1.2.9 Query the Machine's volume collection ..................................................... 22
  1.3 Defining and using Machine Templates ........................................................... 23
    1.3.1 Retrieve the CEP ........................................................................................... 23
    1.3.2 Create a new Machine Template ............................................................... 23
    1.3.3 Create a new Machine by using a Machine Template ................................ 24
  1.4 Creating a new Machine connected to an existing Volume ............................ 24
    1.4.1 Retrieve the CEP ........................................................................................... 24
    1.4.2 Get the list of Volumes ............................................................................... 25
    1.4.3 Choose a Volume ......................................................................................... 25
    1.4.4 Create a new Machine ................................................................................ 25
    1.4.5 Query new Machine .................................................................................... 26
  1.5 Defining and using System Templates ............................................................. 27
    1.5.1 Retrieve the CEP ........................................................................................... 27
    1.5.2 Create a new System Template .................................................................. 27
    1.5.3 Create a new System by using a System Template .................................... 29
    1.5.4 Query the new System ............................................................................... 30
  1.6 Editing System Templates .................................................................................. 30
    1.6.1 Edit an existing System Template ............................................................... 30
    1.6.2 Create a new System using a System Template ........................................ 33
    1.6.3 Query the new System ............................................................................... 33
  1.7 Provider responses and return values for special cases .................................. 34
    1.7.1 Unrecognized attributes in request filters .................................................. 34
    1.7.2 Unreasonable requests ............................................................................... 34
  2 Resources modeling aspects .................................................................................. 34
    2.1 Conventions ....................................................................................................... 34
    2.2 1-to-N relationships using arrays and collections .......................................... 35
Cloud Infrastructure Management Interface (CIMI) Primer

3 Groups of related Resources and their serialization

3.1 The Cloud Entry Point (CEP) ........................................ 40
3.2 Machine Resources group ............................................. 41
  3.2.1 Machine Resources and their relationships .................... 41
  3.2.2 Machine serialization ........................................... 43
  3.2.3 DiskCollection serialization ..................................... 45
  3.2.4 MachineCollection serialization .................................. 46
  3.2.5 MachineTemplate serialization .................................... 48
  3.2.6 MachineTemplateCollection serialization ......................... 51
  3.2.7 MachineConfiguration serialization ............................... 52
  3.2.8 MachineConfigurationCollection serialization .................... 53
  3.2.9 MachineImage serialization ....................................... 54
  3.2.10 MachineImageCollection serialization ............................ 55

3.3 Credential Resources group ........................................ 56
  3.3.1 Credential serialization .......................................... 56
  3.3.2 CredentialCollection serialization ............................... 57
  3.3.3 CredentialTemplate serialization .................................. 58
  3.3.4 CredentialTemplateCollection serialization ....................... 59

3.4 Volume Resources group ............................................. 61
  3.4.1 Volume Resources group and their relationships .................... 61
  3.4.2 Volume serialization ............................................... 61
  3.4.3 VolumeCollection serialization .................................... 63
  3.4.4 locatedVolumeCollection serialization ............................. 64
  3.4.5 VolumeTemplate serialization ...................................... 65
  3.4.6 VolumeTemplateCollection serialization ........................... 67
  3.4.7 VolumeConfiguration serialization ................................ 68
  3.4.8 VolumeConfigurationCollection serialization ....................... 69
  3.4.9 VolumeImage serialization ......................................... 70
  3.4.10 VolumeImageCollection serialization ............................... 71

3.5 System Resources group ............................................. 72
  3.5.1 System Resources and their relationships ........................ 72
  3.5.2 System serialization ................................................ 74
  3.5.3 SystemCollection serialization ...................................... 76
  3.5.4 SystemService serialization ....................................... 77
  3.5.5 SystemTemplate serialization ...................................... 79
  3.5.6 SystemTemplateCollection serialization ............................ 81

3.6 Network Resources group ............................................ 83
  3.6.1 Network Resources and their relationships ........................ 83
  3.6.2 Network serialization ............................................. 85
  3.6.3 NetworkCollection serialization .................................... 87
  3.6.4 NetworkTemplate serialization ..................................... 88
  3.6.5 NetworkTemplateCollection serialization ........................... 90
  3.6.6 NetworkService serialization ...................................... 91
  3.6.7 NetworkServiceCollection serialization ........................... 92
  3.6.8 NetworkServiceTemplate serialization .............................. 94
  3.6.9 NetworkServiceTemplateCollection serialization .................... 96
  3.6.10 ProtocolSegment serialization ..................................... 97
  3.6.11 ProtocolSegmentCollection serialization .......................... 98
  3.6.12 ProtocolSegmentTemplate serialization ............................ 99
  3.6.13 ProtocolSegmentTemplateCollection serialization .................. 101
  3.6.14 ProtocolEndpoint serialization ................................... 102
  3.6.15 ProtocolEndpointCollection serialization .......................... 104
  3.6.16 ProtocolEndpointTemplate serialization ............................ 105
Figures

170 Figure 1 - Graphical conventions ........................................................................................................35
171 Figure 2 – Array-based 1-n Resource relationship from A to Bs .........................................................36
172 Figure 3 – Collection-based 1-n Resource relationship from A to Bs ..................................................37
173 Figure 4 – Template instantiation mechanism .....................................................................................38
174 Figure 5 – Cloud Entry Point resources ...............................................................................................41
175 Figure 6 – Machine Resources group ..................................................................................................42
176 Figure 7 – Volume Resources group ....................................................................................................61
177 Figure 8 – System Resources group .....................................................................................................73
178 Figure 9 – SystemTemplate Resources group .....................................................................................74
179 Figure 10 – Network Resources group ................................................................................................83
180 Figure 11 – Network Resources group: Resource structure relationships .............................................84
181 Figure 12 – Network Resources group: Network connectivity perspective .............................................85
182 Figure 13 – Monitoring Resources group ............................................................................................113
183
Foreword

The *Cloud Infrastructure Management Interface (CIMI) Primer* (DSP2027) was prepared by the Cloud Management Working Group of the DMTF. This document contains scenarios that describe common uses of the CIMI protocol.

This specification has been developed as a result of joint work with many individuals and teams, including:

- Diez Köper, Fujitsu (Editor)
- Larry Lamers, VMware (Editor)

**Contributors:**

- Hemal Shah, Broadcom
- John Crandall, Brocade Communications Systems
- Paul Lipton, CA Technologies
- Efraim Moscovich, CA Technologies
- Mark Carlson, DMTF Fellow
- Marvin Waschke, DMTF Fellow
- Steven Neely, Cisco
- Shishir Pardikar, Citrix Systems Inc.
- George Ericson, EMC
- Norbert Floeren, Ericsson AB
- Arturo Martin de Nicolas, Ericsson AB
- Ruby Krishnaswamy, France Telecom Group
- Jacques Durand, Fujitsu
- Kazunori Iwasa, Fujitsu
- Jesus Molina, Fujitsu
- Tom Rutt, Fujitsu
- Maarten Wiggers, Fujitsu
- Bryan Murray, Hewlett
- Derek Coleman, Hewlett Packard Company
- Robert Freund, Hitachi, Ltd.
- Fred Maciel, Hitachi, Ltd.
- Eric Wells, Hitachi, Ltd.
- Zhexuan Song, Huawei
- Jeff Wheeler, Huawei
- Aaron Zhang, Huawei
- Heng Liang Zhang, Huawei
- Doug Davis, IBM
- Mark Johnson, IBM
- Andreas Maier, IBM
- Matthew Rutkowski, IBM
- Keith Bankston, Microsoft Corporation
- Nathan Burkhart, Microsoft Corporation
- Josh Cohen, Microsoft Corporation
- Yigal Edery, Microsoft Corporation
- Colleen Evans, Microsoft Corporation
- Krishnan Gopalan, Microsoft Corporation
- John Parchem, Microsoft Corporation
- Nihar Shah, Microsoft Corporation
- Hiroshi Dempo, NEC Corporation
- James Livingston, NEC Corporation
- Ryuichi Ogawa, NEC Corporation
- Steve Carter, Novell
- Ashok Malhotra, Oracle
<table>
<thead>
<tr>
<th>Page</th>
<th>Name</th>
<th>Company</th>
</tr>
</thead>
<tbody>
<tr>
<td>236</td>
<td>Gilbert Pilz</td>
<td>Oracle</td>
</tr>
<tr>
<td>237</td>
<td>Jack Yu</td>
<td>Oracle</td>
</tr>
<tr>
<td>238</td>
<td>Marios Andreou</td>
<td>Red Hat</td>
</tr>
<tr>
<td>239</td>
<td>David Lutterkort</td>
<td>Red Hat</td>
</tr>
<tr>
<td>240</td>
<td>Steve Winkler</td>
<td>SAP AG</td>
</tr>
<tr>
<td>241</td>
<td>Enrico Ronco</td>
<td>Telecom Italia</td>
</tr>
<tr>
<td>242</td>
<td>Federico Rossini</td>
<td>Telecom Italia</td>
</tr>
<tr>
<td>243</td>
<td>Fernando de la Iglesia</td>
<td>Telefónica</td>
</tr>
<tr>
<td>244</td>
<td>Fermín Galán</td>
<td>Telefónica</td>
</tr>
<tr>
<td>245</td>
<td>Miguel Peñalvo</td>
<td>Telefónica</td>
</tr>
<tr>
<td>246</td>
<td>Alvaro Polo</td>
<td>Telefónica</td>
</tr>
<tr>
<td>247</td>
<td>Alan Sill</td>
<td>Texas Tech University</td>
</tr>
<tr>
<td>248</td>
<td>Vince Lubsey</td>
<td>Virtustream Inc.</td>
</tr>
<tr>
<td>249</td>
<td>Winston Bumpus</td>
<td>VMware Inc.</td>
</tr>
<tr>
<td>250</td>
<td>Jim Davis</td>
<td>WBEM Solutions</td>
</tr>
<tr>
<td>251</td>
<td>Ghazanfar Ali</td>
<td>ZTE Corporation</td>
</tr>
<tr>
<td>252</td>
<td>Junsheng Chu</td>
<td>ZTE Corporation</td>
</tr>
<tr>
<td>253</td>
<td>Bhumip Khasnabish</td>
<td>ZTE Corporation</td>
</tr>
</tbody>
</table>
Introduction

For the sake of simplicity, in each of the following scenarios, the Cloud provider only supports the minimum features needed to demonstrate the features highlighted by each scenario. Therefore, the results of the query to the Cloud Entry Point (CEP) to retrieve the list of supported features are customized for each scenario. Additionally, the URI of the Cloud Entry Point is assumed to be http://example.com/CEP and all resources are assumed to be available in the example.com domain and accessible with the same protocol (HTTP). In the HTTP request-response examples in this document, the creation of the connection and HTTP headers not mandated by the CIMI specification are omitted for brevity.
1 Common scenarios of consumer-provider interactions

1.1 Creating a new Machine

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

1.1.1 Retrieve the CEP

The CEP provides 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 /CEP HTTP/1.1

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

{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/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" }
}

1.1.2 Retrieve the list of Machine Images

Before you can create a new Machine, first decide what kind of operating system or software you want to have preinstalled. The Machine Images collection is the set of Machine Images that this Cloud offers - note that some Machine Images may be predefined by the Cloud while some may be user created. The URI path comes from the data returned in the query to CEP for the machineImages key.

To retrieve the list of Machine Images, use the following syntax:

GET /machineImages HTTP/1.1

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

{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/MachineImageCollection",
  "id": "http://example.com/machineImages",
  "updated": "2015-01-25T12:00:00Z",
  "parent": "http://example.com/CEP",
  "count": 3,
  "machineImages": [
    {
      "resourceURI": "http://schemas.dmtf.org/cimi/2/MachineImage",
      "id": "http://example.com/images/WinXP-SP2",
      "name": "WinXP SP2",
      "description": "Windows XP with Service Pack 2",
      "created": "2015-01-15T12:00:00Z",
      "updated": "2015-01-15T12:00:00Z",
      "parent": "http://example.com/machineImages",
      "imageLocation": "http://example.com/data/8934322"
    },
    // More MachineImages...
  ]
}
1.1.3 Choose a Machine Image

Next examine each Machine Image to find one that meets your needs. The first one is acceptable, so it will be used 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, the following syntax could have been used instead:

```
GET /machineImages?$first=1&$last=1 HTTP/1.1
```

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

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/MachineImageCollection",
 "id": "http://example.com/machineImages",
 "updated": "2015-01-25T12:00:00Z",
 "parent": "http://example.com/CEP",
 "count": 3,
 "machineImages": [
 { "resourceURI": "http://schemas.dmtf.org/cimi/2/MachineImage",
  "id": "http://example.com/images/WinXP-SP2",
  "name": "WinXP SP2",
  "description": "Windows XP with Service Pack 2",
  "created": "2015-01-01T12:00:00Z",
  "updated": "2015-01-01T12:00:00Z",
  "parent": "http://example.com/machineImages",
  "imageLocation": "http://example.com/data/8934322"
 } ]
}
```

Note that you do not need to specify $first=1 in this case because "1" is its default value. The first machineImage is returned.

1.1.4 Retrieve the list of Machine Configurations

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

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

Published Version 2.0.0
HTTP/1.1 200 OK
Content-Type: application/json

```
{  "resourceURI": "http://schemas.dmtf.org/cimi/2/MachineConfigurationCollection",
  "id": "http://example.com/machineConfigs",
  "updated": "2015-01-25T12:00:00Z",
  "parent": "http://example.com/CEP",
  "count": 3,
  "machineConfigurations": [
    {  "resourceURI": "http://schemas.dmtf.org/cimi/2/MachineConfiguration",
      "id": "http://example.com/machineConfigs/tiny",
      "name": "tiny",
      "description": "a teenie tiny one",
      "created": "2015-01-01T12:00:00Z",
      "updated": "2015-01-01T12:00:00Z",
      "parent": "http://example.com/machineConfigs",
      "cpu": 1,
      "memory": 4000000,
      "disks": [  
        { "capacity": 50000000 }
      ]
    },
    {  "resourceURI": "http://schemas.dmtf.org/cimi/2/MachineConfiguration",
      "id": "http://example.com/machineConfigs/small",
      "name": "small",
      "description": "a small sized one",
      "created": "2015-01-01T12:00:00Z",
      "updated": "2015-01-01T12:00:00Z",
      "parent": "http://example.com/machineConfigs",
      "cpu": 1,
      "memory": 8000000,
      "disks": [  
        { "capacity": 500000000 }
      ]
    },
    {  "resourceURI": "http://schemas.dmtf.org/cimi/2/MachineConfiguration",
      "id": "http://example.com/machineConfigs/medium",
      "name": "medium",
      "description": "a medium one",
      "created": "2015-01-25T12:00:00Z",
      "updated": "2015-01-25T12:00:00Z",
      "parent": "http://example.com/machineConfigs",
      "cpu": 1,
      "memory": 16000000,
      "disks": [  
        { "capacity": 1000000000 },
        { "capacity": 1000000000 }
      ]
    }
  ]
}
```

### 1.1.5 Choose a Machine Configuration

Next examine the returned list and select a Machine Configuration that suits your needs. The first one is acceptable, so it will be used later. It is identified by the `id` "http://example.com/machineConfigs/tiny".
1.1.6 Create a new Credential Resource

You want to use your own `userName` and `password` attributes 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 to where to POST a new Credential resource. To retrieve the Credential resource:

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

The response is:
```
HTTP/1.1 200 OK
Content-Type: application/json

{ "resourceURI": "http://schemas.dmtf.org/cimi/2/CredentialCollection",
  "id": "http://example.com/credentials",
  "operations": [ { "rel": "add", "href": "http://example.com/credentials" } ]
}
```

Notice at this point that 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 CEP:

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

The response is:
```
HTTP/1.1 200 OK
Content-Type: application/json

{ "resourceURI": "http://schemas.dmtf.org/cimi/2/ResourceMetadataCollection",
  "id": "http://example.com/resourceMetadata",
  "updated": "2015-01-25T12:00:00Z",
  "parent": "http://example.com/CEP",
  "count": 1,
  "resourceMetadatas": [
    { "resourceURI": "http://schemas.dmtf.org/cimi/2/ResourceMetadata",
      "id": "http://example.com/resources/Credential",
      "typeURI": "http://schemas.dmtf.org/cimi/2/Credential",
      "name": "Credential",
      "created": "2015-01-01T12:00:00Z",
      "updated": "2015-01-01T12:00:00Z",
      "parent": "http://example.com/resourceMetadata",
      "attributes": [
        { "name": "userName", "namespace": "http://example.com",
          "type": "string", "required": "true" },
        { "name": "password", "namespace": "http://example.com",
          "type": "string", "required": "true" }
      ]
    }
  ]
}
```

Now iterate over the list of `resourceMetadata` entries in the collection for the one whose “typeURI” is “http://schemas.dmtf.org/cimi/2/Credential”. After you find it, you can now examine the extensions this provider has added to the Credential resource. The above indicates that the Credential resource has been extended and must include two attributes called “userName” 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/2/CredentialCreate",
   "name": "Default",
   "description": "My Default User",
   "credentialTemplate": {
      "username": "JoeSmith",
      "password": "letmein"
   }
}
```

HTTP/1.1 201 Created
Location: http://example.com/creds/12345

NOTE While the "userID" and "password" attributes were discovered via the Credential Resource Metadata, the "name" and "description" attributes are part of the common set of attributes available on all resources. In a future scenario it is shown how the client knew that "userID" and "password" were the proper attribute names for this image type and Cloud provider.

### 1.1.7 Create a new Machine

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

```
GET /machines HTTP/1.1

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

{  "resourceURI": "http://schemas.dmtf.org/cimi/2/MachineCollection",
   "id": "http://example.com/machines",
   "updated": "2015-01-25T12:00:00Z",
   "parent": "http://example.com/CEP",
   "count": 0,
   "operations": [ { "rel": "add", "href": "http://example.com/machines" } ]
}
```

If you only want to know the available operations, issue the following command.

```
GET /machines?$select=operations HTTP/1.1

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

{  "resourceURI": "http://schemas.dmtf.org/cimi/2/MachineCollection",
   "operations": [ { "rel": "add", "href": "http://example.com/machines" } ]
}
```

Now create a new one:

```
POST /machines HTTP/1.1
Content-Type: application/json

{  "resourceURI": "http://schemas.dmtf.org/cimi/2/MachineCreate",
   "name": "myMachine1",
   "description": "My very first machine",
   "machineTemplate": {
      "machineConfig": { "href": "http://example.com/machineConfigs/tiny" },
      "machineImage": { "href": "http://example.com/images/WinXP-SP2" },
      "credential": { "href": "http://example.com/creds/12345" }
   }
}
```
HTTP/1.1 201 Created
Location: http://example.com/machines/843752

The response returns a unique machine reference "http://example.com/machines/843752" that is used in the following subclause.

1.1.8  Query 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/json

```
{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/Machine",
  "id": "http://example.com/machines/843752",
  "name": "myMachine1",
  "description": "My very first machine",
  "created": "2015-08-15T12:15:00Z",
  "updated": "2015-08-15T12:15:00Z",
  "parent": "http://example.com/machines",
  "state": "STOPPED",
  "cpu": 1,
  "memory": 4000000,
  "disks": { "href": "http://example.com/machines/843752/disks" },
  "interfaces": { "href": "http://example.com/machines/843752/NIs" },
}
```

Notes:

- The "state" attribute on the Machine is "STOPPED" because that is the initial state of a new machine.
- The parent attribute refers here to the machines CEP collection, as this Machine was created by directly using the CEP machines collection as a factory. If a Machine was created as part of the creation of a System, the parent attribute would refer to the machines collection inside the System.
- In this simple example there is no resourceMetadata attribute (an attribute optional to support).

1.1.9  Start a Machine

The presence of the "start" operation in the "operations" array of the Machine representation indicates not only the URI to which to POST the "start" operation, 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/2/Action",
  "action": "http://schemas.dmtf.org/cimi/2/action/start"
}
```

HTTP/1.1 204 No Content
1.1.10 Query a Machine

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/2/Machine",
    "id": "http://example.com/machines/843752",
    "name": "myMachine1",
    "description": "My very first machine",
    "created": "2015-08-15T12:15:00Z",
    "updated": "2015-08-15T12:15:00Z",
    "parent": "http://example.com/machines",
    "state": "STARTED",
    "cpu": 1,
    "memory": 4000000,
    "disks": [{ "href": "http://example.com/machines/843752/disks" },
               { "href": "http://example.com/machines/843752/NIs" },
               { "href": "http://schemas.dmtf.org/cimi/2/action/stop" }]
    "operations": [
        { "rel": "edit", "href": "http://example.com/machines/843752" },
        { "rel": "delete", "href": "http://example.com/machines/843752" },
        { "rel": "http://schemas.dmtf.org/cimi/2/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; but rather the "stop" operation is available now instead.

1.1.11 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/2/Action",
    "action": "http://schemas.dmtf.org/cimi/2/action/stop"
}
```

HTTP/1.1 204 No Content

1.1.12 Update a Machine’s attributes

Using a PUT operation on the "edit" operation's URL, you can update some of the attributes 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/2/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 are touched. Because the URL includes the "description" attribute but the HTTP request body does not, that attribute is erased.

### 1.2 Adding a new Volume to a Machine

This scenario creates a new Volume and connects it to an existing Machine.

#### 1.2.1 Obtain the Machine URL

Machine:

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

#### 1.2.2 Retrieve the CEP

The CEP provides the links to the set of resources that are available in this Cloud. Retrieve the CEP to discover the URL to each collection:

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

Response:

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

```

```json
{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/CloudEntryPoint",
  "id": "http://example.com/CEP",
  "baseURI": "http://example.com/",
  "machines": { "href": "http://example.com/machines" },
  "machineConfigs": { "href": "http://example.com/machineConfigs" },
  "machineImages": { "href": "http://example.com/machineImages" },
  "credentials": { "href": "http://example.com/credentials" },
  "volumes": { "href": "http://example.com/volumes" },
  "volumeConfigs": { "href": "http://example.com/volumeConfigs" }
}
```

#### 1.2.3 Get the list of VolumeConfigurations

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

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

Response:

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

```

```json
{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/VolumeConfigurationCollection",
  "id": "http://example.com/volumeConfigs",
  "updated": "2015-08-25T12:00:00Z",
  "parent": "http://example.com/CEP",
  "count": 3,
  "volumeConfigurations": [
    { "resourceURI": "http://schemas.dmtf.org/cimi/2/VolumeConfiguration",
      "id": "http://example.com/volumeConfigs/small",
      "name": "Small",
      "description": "A pretty small one",
      "created": "2015-08-15T12:15:00Z",
      "updated": "2015-08-15T12:15:00Z",
      "parent": "http://example.com/volumeConfigs",
      "type": "http://schemas.dmtf.org/cimi/2/mapped",
      "format": "NTFS",
```
1.2.4 Choose a Volume Configuration

Next examine each Volume Configuration to find the one that meets your needs. The first one is acceptable, so it will be used later.

1.2.5 Create a new Volume

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

```
GET /volumes HTTP/1.1
HTTP/1.1 200 OK
Content-Type: application/json
```

```
{
"resourceURI": "http://schemas.dmtf.org/cimi/2/VolumeCollection",
"id": "http://example.com/volumes",
"updated": "2015-08-25T12:00:00Z",
"parent": "http://example.com/CEP",
"operations": [ { "rel": "add", "href": "http://example.com/volumes" } ]
}
```

Now create a new Volume:

```
POST /volumes HTTP/1.1
Content-Type: application/json
```

```
{
"resourceURI": "http://schemas.dmtf.org/cimi/2/VolumeCreate",
"name": "myVolume1",
"description": "My first new volume",
"volumeTemplate": {
"volumeConfig": { "href": "http://example.com/volumeConfigs/small" }
}
}
```
1.2.6 Retrieve the Volume information

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

GET /volumes/35782 HTTP/1.1

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

```
{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/Volume",
  "id": "http://example.com/volumes/35782",
  "name": "myVolume1",
  "created": "2015-09-15T12:15:00Z",
  "updated": "2015-09-15T12:15:00Z",
  "parent": "http://example.com/volumes",
  "description": "My first new volume",
  "type": "http://schemas.dmtf.org/cimi/2/mapped",
  "capacity": 60000000,
  "operations": [
    { "rel": "edit", "href": "http://example.com/volumes/35782" },
    { "rel": "delete", "href": "http://example.com/volumes/35782" }
  ]
}
```

1.2.7 Retrieve the Machine's Volume collection

Before you can connect this new Volume to your Machine, you first need to retrieve the Machine's Volume collection so that you know to where to send your request. First retrieve the Machine to get the reference to the collection (even if this collection is empty):

GET /machines/843752 HTTP/1.1

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

```
{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/Machine",
  "id": "http://example.com/machines/843752",
  "name": "myMachine1",
  "description": "My very first machine",
  "created": "2015-08-15T12:15:00Z",
  "updated": "2015-08-15T12:15:00Z",
  "parent": "http://example.com/machines",
  "state": "STARTED",
  "cpu": 1,
  "memory": 40000000,
  "disks": { "href": "http://example.com/machines/843752/disks" },
  "volumes": { "href": "http://example.com/machines/843752/volumes" },
  "interfaces": { "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/2/action/stop",
      "href": "http://example.com/machines/843752" }
  ]
}
```

Note that in the previous scenario, the "volumes" attribute was not present due to the limited scope of that scenario; however, now the "volumes" attribute appears because the scenario (and features of our sample provider) are expanded to include support for Volumes.
Now retrieve the Volume collection:

GET /machines/843752/volumes HTTP/1.1

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

```json
{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/locatedVolumeCollection",
  "id": "http://example.com/machines/843752/volumes",
  "updated": "2015-08-15T12:00:00Z",
  "parent": "http://example.com/machines/843752",
  "operations": [
    { "rel": "add", "href": "http://example.com/machines/843752/volumes" },
    { "rel": "insert", "href": "http://example.com/machines/843752/volumes" },
    { "rel": "remove", "href": "http://example.com/machines/843752/volumes" }
  ]
}
```

Note that there are no Volumes currently connected to this Machine.

Alternatively, as an optimization, this collection could have been retrieved at the same time as the original Machine by using the $expand query parameter:

GET /machines/843752?$expand=volumes HTTP/1.1

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

```json
{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/Machine",
  "id": "http://example.com/machines/843752",
  "name": "myMachine1",
  "description": "My very first machine",
  "created": "2015-08-15T12:15:00Z",
  "updated": "2015-08-15T12:15:00Z",
  "parent": "http://example.com/machines",
  "state": "STARTED",
  "cpu": 1,
  "memory": 4000000,
  "disks": [ { "href": "http://example.com/machines/843752/disks" },
    "volumes": [ { "href": "http://example.com/machines/843752/volumes",
      "resourceURI": "http://schemas.dmtf.org/cimi/2/locatedVolumeCollection",
      "id": "http://example.com/machines/843752/volumes",
      "updated": "2015-08-15T12:00:00Z",
      "parent": "http://example.com/machines/843752",
      "count": 0,
      "operations": [ { "rel": "add", "href": "http://example.com/machines/843752/volumes" },
        { "rel": "insert", "href": "http://example.com/machines/843752/volumes" },
        { "rel": "remove", "href": "http://example.com/machines/843752/volumes" }
      ]
    },
    "interfaces": [ { "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/2/action/stop", "href": "http://example.com/machines/843752" }
      ]
    ]
  ]
}
```
1.2.8 Connect the new Volume to a Machine

You connect the Volume to the Machine by using the "insert" operation on the Volume collection and pass in the reference to the new Volume resource:

```
POST http://example.com/machines/843752/volumes HTTP/1.1
Content-Type: application/json
{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/Action",
  "action": "http://schemas.dmtf.org/cimi/2/action/connectvolume",
  "initialLocation": "V",
  "credentials": { "href": "http://example.com/credentials/456" },
  "volume": { "href": "http://example.com/volumes/35782" }
}
```

HTTP/1.1 200 OK
Content-Type: application/json
{
  ...
  a serialization of the updated collection at:
  http://example.com/machines/843752/volumes
  or
  (if the Job Resource is supported) a pointer to a job:
  CIMI-Job-URI: <uri-to-Job>
  That represents the execution of this connection.
  ...
}

1.2.9 Query the Machine’s volume collection

Retrieve the Machine’s volume collection to get the complete list of Volumes and use the list to verify that the update was successful:

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

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

```json
{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/locatedVolumeCollection",
  "id": "http://example.com/machines/843752/volumes",
  "updated": "2015-09-15T12:15:00Z",
  "parent": "http://example.com/machines/843752",
  "count": 1,
  "locatedVolumes": [
    { "resourceURI": "http://schemas.dmtf.org/cimi/2/locatedVolume",
      "id": "http://example.com/volumes/35782",
      "created": "2015-09-15T12:15:00Z",
      "updated": "2015-09-15T12:15:00Z",
      "parent": "http://example.com/volumes",
      "initialLocation": "V",
      "volume": { "href": "http://example.com/volumes/35782" },
      "operations": [
        { "rel": "edit","href": "http://example.com/volumes/35782 "},
        { "rel": "delete","href": "http://example.com/volumes/35782 " }
      ]
    },
    { "rel": "add", "href": "http://example.com/machines/843752/volumes" }
  ]
}
```
1.3 Defining and using Machine Templates

This scenario creates a new Machine Template that is used to create a new Machine. Machine Templates are convenience resources that allow for well-defined descriptions (configuration, image, etc.) of a Machine to be persisted such that it can be reused later. This feature 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. Commonly, Machine Templates are 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, reuse the configuration information already obtained in the previous scenarios.

1.3.1 Retrieve the CEP

The CEP provides the links to the set of resources that are available in this Cloud. Retrieve the CEP to discover the URL to each collection:

```plaintext
GET /CEP HTTP/1.1
HTTP/1.1 200 OK
Content-Type: application/json

{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/CloudEntryPoint",
  "id": "http://example.com/CEP",
  "baseURI": "http://example.com/",
  "machines": { "href": "http://example.com/machines" },
  "machineTemplates": { "href": "http://example.com/machineTemplates" },
  "machineConfigs": { "href": "http://example.com/machineConfigs" },
  "machineImages": { "href": "http://example.com/machineImages" },
  "credentials": { "href": "http://example.com/credentials" }
}
```

1.3.2 Create a new Machine Template

From the previous scenarios, you already have the MachineConfiguration, MachineImage, and Credential resources that are reused for this MachineTemplate:

MachineConfiguration:

http://example.com/machineConfigs/tiny

MachineImage:

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

Credential:

http://example.com/creds/12345

Before you can create the new MachineTemplate, you first need to determine the URL to which the POST is sent. This location 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/json

```json
{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/MachineTemplateCollection",
  "id": "http://example.com/machineTemplates",
  "updated": "2015-08-25T12:00:00Z",
  "parent": "http://example.com/CEP",
  "count": 0,
  "operations": [{
    "rel": "add",
    "href": "http://example.com/machineTemplates"
  }]
}
```

Note that there are no MachineTemplates in the environment right now.

Now create the new MachineTemplate resource:

```
POST /machineTemplates HTTP/1.1
Content-Type: application/json

{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/MachineTemplate",
  "name": "Demo1",
  "description": "My first demo",
  "machineConfig": { "href": "http://example.com/machineConfigs/tiny" },
  "machineImage": { "href": "http://example.com/images/WinXP-SP2" },
  "credential": { "href": "http://example.com/creds/12345" }
}
```

HTTP/1.1 201 Created
Location: http://example.com/machineTemplates/82754

1.3.3 Create a new Machine by using a Machine Template

Now create a new Machine by using this Machine Template:

```
POST /machines HTTP/1.1
Content-Type: application/json

{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/MachineCreate",
  "name": "myMachine2",
  "description": "My second machine",
  "machineTemplate": { "href": "http://example.com/machineTemplates/82754" }
}
```

HTTP/1.1 201 Created
Location: http://example.com/machines/843799

1.4 Creating a new Machine connected to an existing Volume

This scenario creates a new Machine that boots from an existing Volume. This simple example assumes that the user knows that there is an existing Volume with the bootable property equal to true.

1.4.1 Retrieve the CEP

The CEP provides the links to the set of resources that are available in this Cloud. Retrieve the CEP to discover the URL to each collection:

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

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

```json
```
1018 }

1.4.2 Get the list of Volumes

When you create a new Machine from a Volume, you need to decide which Volume to use. The Volume collection is the set of existing Volumes that this Cloud offers:

GET /volumes HTTP/1.1

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

{
  "resourceType": "http://schemas.dmtf.org/cimi/2/VolumeCollection",
  "id": "http://example.com/volumes",
  "created": "2015-08-25T12:00:02Z",
  "updated": "2015-08-25T12:00:02Z",
  "parent": "http://example.com/CEP",
  "count": 2,
  "volumes": [
    {
      "resourceType": "http://schemas.dmtf.org/cimi/2/Volume",
      "id": "http://example.com/volumes/vol1",
      "name": "Win7-Bootable",
      "created": "2015-08-15T12:15:02Z",
      "updated": "2015-08-15T12:15:02Z",
      "parent": "http://example.com/volumes",
      "description": "A bootable volume running Windows 7",
      "state": "AVAILABLE",
      "capacity": 60000000,
      "bootable": true
    },
    {
      "resourceType": "http://schemas.dmtf.org/cimi/2/Volume",
      "id": "http://example.com/volumes/vol2",
      "name": "Generic Volume",
      "created": "2015-08-15T12:15:02Z",
      "updated": "2015-08-15T12:15:02Z",
      "parent": "http://example.com/volumes",
      "description": "A generic volume for Windows",
      "state": "AVAILABLE",
      "capacity": 60000000,
      "bootable": true
    }
  ]
}

1.4.3 Choose a Volume

Next examine each Volume to find the one that meets your needs. The first one is acceptable, so it will be used later.

1.4.4 Create a new Machine

Retrieve the Machines collection so you know to where to POST a new Machine:
GET /machines HTTP/1.1
HTTP/1.1 200 OK
Content-Type: application/json

{
    "resourceType": "http://schemas.dmtf.org/cimi/2/MachineCollection",
    "id": "http://example.com/machines",
    "updated": "2015-08-25T12:00:00Z",
    "parent": "http://example.com/CEP",
    "count": 0,
    "operations": [
        { "rel": "add", "href": "http://example.com/machines" }
    ]
}

Now create a new one, connecting it to the bootable Volume:

POST /machines HTTP/1.1
Content-Type: application/json

{
    "resourceType": "http://schemas.dmtf.org/cimi/2/MachineCreate",
    "name": "myMachine2",
    "description": "My second machine",
    "machineTemplate": {
        "volumes": [
            { "initialLocation": "V",
              "href": "http://example.com/volumes/voll" }
        ]
    }
}

Note that the MachineTemplate in this case does not specify a MachineImage or MachineConfiguration to use. In this example, for simplicity, you can assume that the provider has default values for those.

HTTP/1.1 201 Created
Location: http://example.com/machines/852108

1.4.5 Query new Machine

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

GET /machines/852108 HTTP/1.1
HTTP/1.1 200 OK
Content-Type: application/json

{
    "resourceType": "http://schemas.dmtf.org/cimi/2/Machine",
    "id": "http://example.com/machines/852108",
    "name": "myMachine2",
    "description": "My second machine",
    "created": "2015-03-26T10:04:00Z",
    "updated": "2015-03-26T10:04:00Z",
    "parent": "http://example.com/machines",
    "state": "STOPPED",
}
Notice the "state" attribute on the Machine is "STOPPED", which is the initial state of a new machine.

1.5 Defining and using System Templates

This scenario creates a new System Template that is used to create a new System. System Templates are convenience resources that allow for well-defined descriptions (configuration, image, etc.) of a System to be persisted such that it can be reused later.

1.5.1 Retrieve the CEP

The CEP provides the links to the set of resources that are available in this Cloud. Retrieve the CEP to discover the URL to each collection:

```
GET /CEP HTTP/1.1
HTTP/1.1 200 OK
Content-Type: application/json
{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/CloudEntryPoint",
  "id": "http://example.com/CEP",
  "baseURI": "http://example.com/",
  "systemTemplates": { "href": "http://example.com/systemTemplates" },
  "machineTemplates": { "href": "http://example.com/machineTemplates" },
  "credentialTemplates": { "href": "http://example.com/credentialTemplates" },
  "volumeTemplates": { "href": "http://example.com/volumeTemplates" }
}
```

1.5.2 Create a new System Template

A SystemTemplate is defined so that when instantiated the result is a Machine is created, a Volume is connected to the Machine, and a Credential resource exists. To achieve this configuration, the following component Resources are included: a SystemTemplate definition, a MachineTemplate by value, a VolumeTemplate by reference, and a CredentialTemplate by reference. The VolumeTemplate and CredentialTemplate resources are already available:

VolumeTemplate:

http://example.com/volumeTemplates/95839

CredentialTemplate:

http://example.com/credentialTemplates/72000
Note Alternatively, the VolumeTemplate and CredentialTemplate may be included by value in the MachineTemplate definition below. However, it is beneficial to immediately see in the SystemTemplate the resources that are involved and in general, automatic creation of the credential is more secure.

Before creating a SystemTemplate, the URL to which the POST is sent needs to be determined. This location is obtained from the SystemTemplate collection URL that was returned as part of the CEP.

GET /systemTemplates HTTP/1.1
HTTP/1.1 200 OK
Content-Type: application/json
{
    "resourceURI": "http://schemas.dmtf.org/cimi/2/SystemTemplateCollection",
    "id": "http://example.com/systemTemplates",
    "updated": "2015-08-25T12:00:00Z",
    "parent": "http://example.com/CEP",
    "count": 0,
    "operations": [
        { "rel": "add", "href": "http://example.com/systemTemplates" }
    ]
}

Now create the new System Template resource:

POST /systemTemplates HTTP/1.1
Content-Type: application/json
{
    "resourceURI": "http://schemas.dmtf.org/cimi/2/SystemTemplate",
    "name": "System Demo1",
    "description": "My first system template demo",
    "componentDescriptors": [
        { "name": "MyMachine",
          "type": "http://schemas.dmtf.org/cimi/2/Machine",
          "machineTemplate":
            { "name": "Machine in system demo",
              "description": "Machine in system",
              "machineConfig": { "href": "http://example.com/machineConfigs/tiny" },
              "machineImage": { "href": "http://example.com/images/WinXP-SP2" },
              "credential": { "href": "#MyCredential" },
              "volumes": [
                { "initialLocation": "/vol",
                  "href": "#MyVolume"
                }
              ]
            },
        { "name": "MyCredential",
          "type": "http://schemas.dmtf.org/cimi/2/Credential",
          "credentialTemplate":
            { "href": "http://example.com/credentialTemplates/72000" }
        },
        { "name": "MyVolume",
          "type": "http://schemas.dmtf.org/cimi/2/Volume",
          "volumeTemplate": { "href": "http://example.com/volumeTemplates/95839" }
      ]
}
}
1.5.3 Create a new System by using a System Template

Now create a new System by using this System Template:

```json
POST /systems HTTP/1.1
Content-Type: application/json

{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/SystemCreate",
  "name": "MySystem1",
  "description": "My first system",
  "systemTemplate": {
    "href": "http://example.com/systemTemplates/48920"
  }
}
```

HTTP/1.1 201 Created
Location: http://example.com/systems/78342

Note that, alternatively, the provider could have decided to return a reference to a Job resource instead of waiting until the System is completely created. Instead of the above 201 response, this type of request could have resulted in the following response:

```json
HTTP/1.1 202 Accepted
CIMI-Job-URI: http://example.com/Jobs/90001

{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/Job",
  "id": "http://example.com/Jobs/90001",
  "name": "SystemCreationJob",
  "created": "2015-03-15T12:15:00Z",
  "updated": "2015-03-15T12:15:00Z",
  "parent": "http://example.com/CEP",
  "targetResource": {
    "href": "http://example.com/systems"},
  "affectedResources": {
    "href": "http://example.com/systems/110001",
  },
  "action": "add",
  "status": "RUNNING",
  "progress": 30,
  "timeOfStatusChange": "2015-03-15T12:15:00Z",
  "isCancellable": "true",
  "nestedJobs": [
    {
      "href": "http://example.com/Jobs/90002"
    },
    {
      "href": "http://example.com/Jobs/90003"
    }
  ]
}
```

According to this response, the provider chose to create two nested Jobs and the "affectedResources" attribute includes a reference to the newly created System. Periodic retrieval of the Job's representation...
allows the Consumer to determine when the Job is completed, i.e., it is completed when the "progress" attribute has a value of 100.

1.5.4 Query the new System

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

```
GET /systems/87342
```

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

```
{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/System",
  "id": "http://example.com/systems/78342",
  "name": "MySystem1",
  "description": "My first system",
  "created": "2015-08-15T12:15:00Z",
  "updated": "2015-08-15T12:15:00Z",
  "parent": "http://example.com/systems",
  "state": "STOPPED",
  "machines": {
    "href": "http://example.com/systems/87432/machines"
  },
  "credentials": {
    "href": "http://example.com/systems/87342/creds"
  },
  "volumes": {
    "href": "http://example.com/systems/87342/vols"
  },
  "operations": [
    {"rel": "edit", "href": "http://example.com/systems/78342"
  }
}
```

1.6 Editing System Templates

In this scenario a second Machine is added to an existing System Template.

1.6.1 Edit an existing System Template

Edit the System Template created in a previous scenario and add another machine that shares its credential and volume resources:

```
SystemTemplate:
http://example.com/systemTemplates/48920
```

Retrieve the existing SystemTemplate definition:

```
GET /systemTemplates/48920 HTTP/1.1
```

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

```
{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/SystemTemplate",
  "name": "System Demo1",
  "description": "My first system template demo",
  "created": "2015-08-15T12:15:00Z",
  "updated": "2015-08-15T12:15:00Z",
  "parent": "http://example.com/systemTemplates",
  "componentDescriptors": [
    {"name": "MyMachine",
```
"type": "http://schemas.dmtf.org/cimi/2/Machine",
  "machineTemplate":
    { "name": "Machine in system demo",
      "description": "Machine in system",
      "machineConfig": { "href": "http://example.com/machineConfigs/tiny" },
      "machineImage": { "href": "http://example.com/images/WinXP-SP2" },
      "credential": { "href": "#MyCredential" },
      "volumes": [
        { "initialLocation": "/vol",
          "href": "#MyVolume"
        }
      ]
    },
  
  "name": "MyCredential",
  "type": "http://schemas.dmtf.org/cimi/2/Credential",
  "credentialTemplate":
    { "href": "http://example.com/credentialTemplates/72000" }
},

  "name": "MyVolume",
  "type": "http://schemas.dmtf.org/cimi/2/Volume",
  "volumeTemplate": { "href": "http://example.com/volumeTemplates/95839" }
],

"operations": [
  { "rel": "edit", "href": "http://example.com/systemTemplates/48920" }]
]

Now update the SystemTemplate resource:
PUT /systemTemplates/48920 HTTP/1.1
Content-Type: application/json

{ "resourceURI": "http://schemas.dmtf.org/cimi/2/SystemTemplate",
  "name": "System Demo1",
  "description": "My first system template demo",
  "parent": "http://example.com/systemTemplates",
  "componentDescriptors": [
    { "name": "MyMachine",
      "type": "http://schemas.dmtf.org/cimi/2/Machine",
      "machineTemplate":
        { "name": "Machine in system demo",
          "description": "Machine in system",
          "machineConfig": { "href": "http://example.com/machineConfigs/tiny" },
          "machineImage": { "href": "http://example.com/images/WinXP-SP2" },
          "credential": { "href": "#MyCredential" },
          "volumes": [
            { "initialLocation": "/vol",
              "href": "#MyVolume"
            }
          ]
        }
    }
  ]
HTTP/1.1 200 OK
Content-Type: application/json

{ "resourceURI": "http://schemas.dmtf.org/cimi/2/SystemTemplate",
  "name": "System Demo1",
  "description": "My first system template demo",
  "created": "2015-09-15T12:15:00Z",
  "updated": "2015-09-15T12:15:00Z",
  "parent": "http://example.com/systemTemplates",
  "componentDescriptors": [
    { "name": "MyMachine",
      "type": "http://schemas.dmtf.org/cimi/2/Machine",
      "machineTemplate":
        { "name": "Machine in system demo",
          "description": "Machine in system",
          "machineConfig": { "href": "http://example.com/machineConfigs/tiny" },
          "machineImage": { "href": "http://example.com/images/WinXP-SP2" },
          "credential": { "href": "#MyCredential" },
          "volumes": [
            { "initialLocation": "/vol",
              "href": "#MyVolume"
            }
          ]
        },
        "quantity": 2
    },
    { "name": "MyCredential",
      "type": "http://schemas.dmtf.org/cimi/2/Credential",
      "credentialTemplate":
        { "href": "http://example.com/credentialTemplates/72000" }
    },
    { "name": "MyVolume",
      "type": "http://schemas.dmtf.org/cimi/2/Volume",
      "volumeTemplate": { "href": "http://example.com/volumeTemplates/95839" }
    }
  ]
}
1.6.2 Create a new System using a System Template

Now create a new System using this System Template by sending a POST to the same URL that you used in the previous scenario to create a new System:

```
POST /systems HTTP/1.1
Content-Type: application/json

{  
  "resourceURI": "http://schemas.dmtf.org/cimi/2/SystemCreate",
  "name": "MySystem2",
  "description": "My second system",
  "systemTemplate": { "href": "http://example.com/systemTemplates/48920"  
}
}
```

HTTP/1.1 201 Created
Location: http://example.com/systems/78343

1.6.3 Query the new System

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

```
GET /systems/87343
HTTP/1.1 200 OK
Content-Type: application/json

{  
  "resourceURI": "http://schemas.dmtf.org/cimi/2/System",
  "id": "http://example.com/systems/78342",
  "name": "MySystem2",
  "description": "My second system",
  "created": "2015-10-15T12:15:00Z",
  "updated": "2015-10-15T12:15:00Z",
  "parent": "http://example.com/systems",
  "state": "STOPPED",
  "machines": { "href": "http://example.com/systems/87432/machines" },
  "credentials": { "href": "http://example.com/systems/87432/creds" },
  "volumes": { "href": "http://example.com/systems/87432/vols" },
  "operations": [ 
    { "rel": "edit", "href": "http://example.com/systems/78342"  
  ]
}
```
1.7 Provider responses and return values for special cases

1.7.1 Unrecognized attributes in request filters

A syntax error in the filter expression of a request results in an error being generated. The provider returns a 400 'Bad Request' response to a query with an unrecognized attribute name used in the $filter query parameter.

For example:
GET /machines?$filter=aaa='bbb' HTTP/1.1

The Machine resource has no "aaa" attribute: The recommended action is to return a 400 'Bad Request' because the consumer did not follow $filter's syntax, i.e., "aaa" is not a "resource attribute name".

1.7.2 Unreasonable requests

CIMI consumers are expected to make reasonable requests. CIMI providers are not required to advertise maximum attributes for all resources. A CIMI provider may set limits on the length of attribute values it finds reasonable. It may reject a request it deems unreasonable. This is common practice in web-based protocols today.

These limits may not all be advertised in the ResourceMetadata, although it is recommended that CIMI providers do so. A CIMI provider that receives a request that exceeds any of these limits, returns a response with an appropriate standard HTTP status code, e.g., HTTP return code 413 'Request Entity Too Large'.

2 Resources modeling aspects

2.1 Conventions

The figures of this section do not show the attributes of resources, but only the relationships across resources. In all the figures, some referred resources such as resourceMetadata, meters and eventLog are not represented, as they play the same role for all resources. The following graphical notations apply in the remaining figures of this document:
2.2 1-to-N relationships using arrays and collections

There are two ways in CIMI for representing 1-N relationships between Resources: either by using arrays, or by using collection resources.

1. Array-based relationships. Such relationships are expected to be fairly static, not often updated, and not involving large quantities of Resources. This is expected to be the case in templates. Arrays in templates are not supposed to require special operations (like insert, remove, for Collections) for their updates: such arrays in templates are supposed to have a fairly stable state and may be changed by entire substitutions (HTTP PUT) if required. The ordinality of these arrays (e.g., 0..n vs. 1..n, as well as possible size limits) are to be advertised in ResourceMetadata. Such arrays are also supposed to have a small enough cardinality that they can be sent back all at once when retrieving their container Resource.
2. Collection-based relationships. Such relationships are expected to be conveniently modified, (i.e., not by substituting the entire relationship frame), by fine-grain operations on the Collection Resource such as insert, remove. Such collections may have a large cardinality but then they are not sent back all at once when retrieving their container Resource: only the reference to the Collection shows in the response to a GET resource.
Collection-based 1-n Resource relationship from A to Bs

2.3 Template instantiation

Templates are Resources that are used to generate other Resources. For example, a MachineTemplate is used to generate Machines. It is said here that the Resources generated from a Template are instances of that Template.

A Template may refer to other templates when the Resource it instantiates is composed. For example, a SystemTemplate typically refers to many other Templates that are needed to instantiate different parts of the System Resource.

Also, a Template may refer to existing Resources other than Templates. For example, a MachineTemplate may refer to an existing Volume that the new Machine is expecting to connect to. Every time the MachineTemplate is used, it will create a new Machine that connects to that same Volume.

Figure 4 shows what happens when instantiating a Resource from a Template that uses other Templates as well as other existing Resources.
Template instantiation mechanism

Figure 4 – Template instantiation mechanism

When instantiating the Template for "A" (A_Template), a new "A" Resource is created (A_resource in the Figure 4),

If the Template refers to other Templates – such as B_Template and C_Template in the figure – then new Resources will be instantiated from these sub-templates (here, B_resource2 and C_resource2) every time the top-level Template (here A_Template) is instantiated. However, the existing Resources that were referred to by the Template – here B_resource1 and D_resource(s) – are just reused and referenced by the new Resource (A_resource). This means every time the Template (here A_template) is used to generate a new Resource (here A_resource), this new Resource will refer to the same existing Resources that were used by this Template (here B_resource1 and D_resource(s)): in some way, any existing Resource referred to by a Template is “shared” by all instances of that Template.

Also note that Templates usually only use arrays for 1..N relationships, as such relationships are supposed to be relatively static and do not justify the additional overhead of an intermediate Resource such as a Collection. However, the generated Resources tend to use Collections as these allow for easier updates and are more scalable. The D_resource Resource above illustrates this case: the A_Template is referring to several of such Resources using an array (shown using the array simplified notion), but the instantiated Resource (A_resource) is referring to the same existing Resources (D_resource instances) using a Collection.
2.4 About the nature of dependencies between resources

Resources often relate to each other. One of the most pervasive relationships is the “parent” relationship. The parent attribute present in every resource refers to its parent resource, i.e., another resource that has control over its lifecycle. A resource is said to be a “child component” of its parent. All resources must have a parent. If the resource is created by POSTing to a Collection of resources of this type, the Collection becomes its parent.

The lifecycle of a resource is controlled by its parent. When the parent resource is deleted, all its children resources (and transitively, their children too) are also deleted. For example, if the parent attribute of MachineImage is set to the MachineTemplate resource, then the MachineImage is a child component of the MachineTemplate and will be deleted when its parent (the MachineTemplate) is deleted.

How the parent attribute is set:

Generally there are three ways a resource is created:

1. By POSTing to a Collection that exists directly under the CEP. In this case the Collection resource will be the parent of the new resources. The parent of such a Collection is the CEP itself. The POST contains the description of the new resource, typically a template for it – either by value or by reference.

2. By POSTing to a Collection that is under another existing resource, of which the new resource will be a component, e.g., by POSTing to a machines Collection of an existing System in order to create and add a new Machine to the System. This can be seen as a variant of #1. In this case the parent of the new component resource is the Collection under the composed resource, itself having the composed resource as parent.

3. As the side effect of creating a composed resource. The template of the composed resource would typically refer to a template for the component resource – e.g., a System template that points to a machine Template. The composed resource generally becomes then the parent (or grandparent if a Collection is involved) of the composed resource. Note that the template for the composed resources may also refer to existing external resources (e.g., a Machine template may refer to existing Volumes to be accessible by that Machine).

The parent attribute value of a resource may be changed later by a Consumer only if the access mode of this attribute allows for it. This would be advertised by a Provider in the ResourceMetadata associated with that resource.

The parent attribute must be set wisely for shared resources. If a resource must be shared by several other resources as component – e.g., a canonical Linux image mandated by an employer would be shared by several MachineTemplates - then it would not be wise to pick one as its parent unless there is a clear understanding that this parent should not be deleted as long as the shared resource is needed elsewhere. In such a case, it is better to have the parent of the shared resource (e.g., the canonical Linux MachineImage) set to the machineImages CEP collection.

2.5 Metadata about a resource

Metadata about attributes:

Metadata about attributes of a resource is indicated in a ResourceMetadata resource associated with this resource. Metadata is indicative of such attribute characteristics as the optionality, the access right (read-write, read-only) and possible values for the attribute.

A ResourceMetadata resource may apply to a resource in three ways:
The ResourceMetadata resource is directly associated with this particular resource (see the resourceMetadata common attribute.)

The ResourceMetadata resource associated with the Template used to generate this particular resource (using the genResourceMetadata attribute) – e.g., the ResourceMetadata associated this way with a MachineTemplate will apply to all Machines generated from this template, unless overridden later by a ResourceMetadata associated to this specific Machine (case (i) above).

The ResourceMetadata resource associated by default with this resource type in this CEP. Such ResourceMetadata applies to all instances of this resource type under the CEP, unless overridden by more specific ResourceMetadata instances (see cases (i) or (ii) above).

Metadata about resources:

At resource level, ResourceMetadata will indicate what attributes are supported by the resource. There might be more – or less – attributes than specified in this document. ResourceMetadata will also indicate what associations are or are not supported by a resource. For example, some Machines may not be associated with any Volume at creation time if their MachineTemplate does not support the volumes and volumeTemplates attributes. In other cases, some Machines may be associated with a limited number of Volumes. Such limitations will be indicated by the ResourceMetadata associated with the template (via the genResourceMetadata attribute), which applies to all resources generated from this template.

Optionality of attributes:

Except for some of the “common attributes” (see section 5.7.1 “Common Resource attributes in the specification), the optionality of attributes is governed by their associated metadata, which may vary from Provider to Provider, or from Consumer to Consumer (i.e., may depend on the CEP that a Consumer has access to).

The common attributes that a Provider must support (as stated in the CIMI specification) for any Resource it implements are listed in this JSON serialization excerpt:

```
"id": string,
"name": string,
"description": string,
"parent": string,
"properties": { string: string, + },
```

The common attributes that a Consumer must always include for any Resource it serializes is the id attribute (as this is the only attribute always required to be supported by the Consumer as stated in the CIMI specification).

3 Groups of related Resources and their serialization

3.1 The Cloud Entry Point (CEP)

The CEP Resource is represented below in Figure 5, with other associated Resources:

The Cloud Entry Point has references to a list of Collections as illustrated in Figure 5. It refers to both Template Resources and other Resources. The CEP does not have a Template itself: it is the only Resource that is always supposed to pre-exist to any Consumer access.
3.2 Machine Resources group

3.2.1 Machine Resources and their relationships

The group of Machine-related Resources is represented below, with other associated Resources.
3.2.1.1 **MachineTemplate**

A MachineTemplate refers to several other resources, either other templates (e.g., VolumeTemplate) or regular resources (MachineImage, Volume…). A Template may refer to other Templates for other types of Resources that need be either associated or made components of the new Resource. This is the case here for `volumeTemplates` attribute used to create new Volumes to be associated with the new Machine. However, such associated or component Resources may also be pre-existing Resources, that the Template knows about and refers to. This is the case of the `volumes` attribute here.

The MachineTemplate supports both cases:

- When a MachineTemplate refers to a Resource such as a Volume, it means that this Volume as an already created Resource will be connected to the new Machine. When using the Template several times, each new Machine will be connected to the same Volume.

- When a MachineTemplate refers to a Resource such as a VolumeTemplate, it means this VolumeTemplate is used every time a new Machine is created, to create a new Volume to be connected to the new Machine. When using the Template several times, each new Machine will be connected to the a new Volume created from the VolumeTemplate.

If the MachineTemplate refers to both VolumeTemplates and existing Volumes, the resulting machine will have a `volumes` attribute referring to a Collection containing both pre-existing Volumes and new Volumes.
The ordinality associated with each referenced resource is shown in the above figure with a typical value or range. The actual ordinality supported by a Provider is determined by a ResourceMetadata resource associated with the MachineTemplate resource type. For example, there is typically one MachineConfiguration resource and one MachineImage resource referred by the MachineTemplate, but it maybe that a provider makes the machineConfig attribute optional. Same for the credential attribute, or the volumes attribute.

MachineTemplate has one-to-many relationships to Volume, as well as to VolumeTemplate and NetworkInterfaceTemplate. Such relationships are typically represented by arrays in templates, as these are simpler than Collection resources although not as easily updated. Machine

The Machine resource has relationships to other resources. Unlike for a template, one-to-many relationships are represented with Collection resources acting as intermediaries between the resource and its sets of referred resources. Again, the ordinality shown here is indicative of the most common cases.

### 3.2.2 Machine serialization

The following pseudo-schemas (see notation in CIMI 1.3) describe the serialization of the Resource in both JSON and XML:

**JSON media type:** application/json

**JSON serialization:**

```json
{
    "resourceURI": "http://schemas.dmtf.org/cimi/2/Machine",
    "id": string,
    "name": string,
    "description": string,
    "created": string,
    "updated": string,
    "parent": string,
    "properties": { string: string, + },
    "vscope": [ valueScope, * ],
    "state": string,
    "cpu": number,
    "memory": number,
    "disks": { "href": string },
    "cpuArch": string,
    "cpuSpeed": number,
    "volumes": { "href": string },
    "interfaces": { "href": string },
    "latestSnapshot": { "href": string },
    "snapshots": { "href": string },
    "meters": { "href": string },
    "eventLog": { "href": string },
    "operations": [
        { "rel": "edit", "href": string, ("available": boolean) },
```

```json
```
XML media type: application/xml

XML serialization:

```xml
<Machine xmlns="http://schemas.dmtf.org/cimi/2">
  <id>xs:anyURI</id>
  <name>xs:string</name>
  <description>xs:string</description>
  <created>xs:dateTime</created>
  <updated>xs:dateTime</updated>
  <parent>xs:anyURI</parent>
  <properties>
    <property key="xs:string">xs:string</property> *
  </properties>
  <vscope>valueScope</vscope>
  <state>xs:string</state>
  <cpu>xs:integer</cpu>
  <memory>xs:integer</memory>
  <disks href="xs:anyURI"/>
  <cpuArch>xs:string</cpuArch>
  <cpuSpeed>xs:integer</cpuSpeed>
  <volumes href="xs:anyURI"/>
  <interfaces href="xs:anyURI"/>
  <latestSnapshot href="xs:anyURI"/>
  <snapshots href="xs:anyURI"/>
  <meters href="xs:anyURI"/>
</Machine>
```
3.2.3 DiskCollection serialization

In the following serializations, the Disk resource is expanded: each item of the Collection shows the Disk attributes, instead of a reference.

JSON serialization:

```json
{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/DiskCollection",
  "id": string,
  "updated": string,
  "parent": string,
  "count": number,
  "disks": [
    {
      "resourceURI": "http://schemas.dmtf.org/cimi/2/Disk",
      "id": string,
      "name": string,
      "description": string,
      "created": string,
      "updated": string,
      "properties": { string: string, + },
      "capacity": number,
      "initialLocation": string,
    }
  ]
}
```
"operations": [
  { "rel": "edit", "href": "string" },
  { "rel": "delete", "href": "string" }
],
...,
"operations": [
  { "rel": "add", "href": "string" }
]
...}

XML serialization:

<Collection resourceURI="http://schemas.dmtf.org/cimi/2/DiskCollection"
  xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <updated> xs:dateTime </updated>
  <parent> xs:anyURI </parent>
  <count> xs:integer </count>
  <disks>
    <Disk>
      <id> xs:anyURI </id>
      <name> xs:string </name>
      <description> xs:string </description>
      <created> xs:dateTime </created>
      <updated> xs:dateTime </updated>
      <property key="xs:string"> xs:string </property> *
      <capacity> xs:integer </capacity>
      <initialLocation> xs:string </initialLocation>
      <operation rel="edit" href="xs:anyURI"/>
      <operation rel="delete" href="xs:anyURI"/>
      <xs:any>*
    </Disk> *
    <disks>
      <operation rel="add" href="xs:anyURI"/>
      <xs:any>*
  </Collection>

3.2.4 MachineCollection serialization

The following pseudo-schemas (see notation in CIMI 1.3) describe the serialization of the Resource in both JSON and XML:
JSON serialization:

```json
{
    "resourceURI": "http://schemas.dmtf.org/cimi/2/MachineCollection",
    "id": string,
    "updated": string,
    "parent": string,
    "count": number,
    "machines": [
        {
            "resourceURI": "http://schemas.dmtf.org/cimi/2/Machine",
            "id": string,
            "name": string,
            "description": string,
            "created": string,
            "updated": string,
            "parent": string,
            "properties": { string: string, + },
            "machine": { "href": string },
            "operations": [
                { "rel": "edit", "href": string },
                { "rel": "delete", "href": string }
            ], +
        }, +
    ],
    "operations": [
        { "rel": "add", "href": string },
        { "rel": "insert", "href": string },
        { "rel": "remove", "href": string }
    ]
}
```

XML serialization:

```xml
<Collection resourceURI="http://schemas.dmtf.org/cimi/2/MachineCollection"
            xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <updated> xs:dateTime </updated>
  <parent> xs:anyURI </parent>
  <count> xs:integer </count>
  <machines>
    <Machine>
      <id> xs:anyURI </id>
      <name> xs:string </name>
    </Machine>
  </machines>
</Collection>
```
3.2.5 MachineTemplate serialization

The following pseudo-schemas (see notation in CIMI 1.3) describe the serialization of the Resource in both JSON and XML:

**JSON media type:** application/json

**JSON serialization:**

```json
{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/MachineTemplate",
  "id": string,
  "name": string,
  "description": string,
  "created": string,
  "updated": string,
  "parent": string,
  "properties": { string: string, + },
  "vscope": [ valueScope, * ],
  "initialState": string,
  "machineConfig": {
    "href": string | ... MachineConfiguration attributes ...
  }
}
```
"machineImage": {
  "href": string | ... MachineImage attributes ...
},
"credential": {
  "href": string | ... CredentialTemplate attributes ...
},
"volumes": [
  { "initialLocation": string, "href": string }, +
],
"volumeTemplates": [
  { "initialLocation": string,
    "href": string,
    ... VolumeTemplate attributes ...
  }, +
],
"interfaceTemplates": [
  { "href": string,
    ... NetworkInterfaceTemplate attributes ...
  }, *
],
"userData": string,
"meterTemplates": [
  { "href": string,
    ... MeterTemplate attributes ...
  }, *
],
"eventLogTemplate": {
  "href": string,
  ... EventLogTemplate attributes ...
},
"genResourceMetadata": { "href": string }
"operations": [
  { "rel": "edit", "href": string },
  { "rel": "delete", "href": string }
]
Cloud Infrastructure Management Interface (CIMI) Primer

XML media type: application/xml

XML serialization:

```xml
<MachineTemplate xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <name> xs:string </name>
  <description> xs:string </description>
  <created> xs:dateTime </created>
  <updated> xs:dateTime </updated>
  <parent> xs:anyURI </parent>
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties>
  <vscope> valueScope </vscope>
  <initialState> xs:string </initialState>
  <machineConfig href="xs:anyURI">
    ... MachineConfiguration attributes ...
  </machineConfig>
  <machineImage href="xs:anyURI">
    ... MachineImage attributes ...
  </machineImage>
  <credential href="xs:anyURI">
    ... CredentialTemplate attributes ...
  </credential>
  <volumes>
    <volume initialLocation="xs:string" href="xs:anyURI" /> *
  </volumes>
  <volumeTemplates>
    <volumeTemplate initialLocation="xs:string" href="xs:anyURI">
      ... VolumeTemplate attributes ...
    </volumeTemplate> *
  </volumeTemplates>
  <interfaceTemplates>
    <interfaceTemplate href="xs:anyURI">
      ... NetworkInterfaceTemplate attributes ...
    </interfaceTemplate> *
  </interfaceTemplates>
  <userData> xs:string </userData>
  <meterTemplates>
    <meterTemplate href="xs:anyURI">
      ... MeterTemplate attributes ...
    </meterTemplate> *
  </meterTemplates>
</MachineTemplate>
```
3.2.6 **MachineTemplateCollection serialization**

The following pseudo-schemas (see notation in CIMI 1.3) describe the serialization of the Resource in both JSON and XML:

This Resource shall be serialized as follows:

**JSON serialization:**

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/MachineTemplateCollection",
 "id": string,
 "updated": string,
 "parent": string,
 "count": number,
 "machineTemplates": [
  { "resourceURI": "http://schemas.dmtf.org/cimi/2/MachineTemplate",
    "id": string,
    ... remaining MachineTemplate attributes ...
  }, +
 ],
 "operations": [ [ "rel": "add", "href": string ] ]
 ...}
```

**XML serialization:**

```
<Collection
   resourceURI="http://schemas.dmtf.org/cimi/2/MachineTemplateCollection"
   xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <updated> xs:dateTime </updated>
  <parent> xs:anyURI </parent>
  <count> xs:integer </count>
```
3.2.7 MachineConfiguration serialization

The following pseudo-schemas (see notation in CIMI 1.3) describe the serialization of the Resource in both JSON and XML:

**JSON media type:** application/json

**JSON serialization:**

```json
{
    "resourceURI": "http://schemas.dmtf.org/cimi/2/MachineConfiguration",
    "id": string,
    "name": string,
    "description": string,
    "created": string,
    "updated": string,
    "parent": string,
    "properties": {
        string: string,
    },
    "vscope": {
        valueScope,
    },
    "cpu": number,
    "memory": number,
    "disks": {
        "capacity": number,
        "format": string,
        "initialLocation": string,
    },
    "cpuArch": string,
    "cpuSpeed": number,
    "operations": {
        "rel": "edit", "href": string,
        "rel": "delete", "href": string
    }
}...```
XML media type: application/xml

XML serialization:

```xml
<MachineConfiguration xmlns="http://schemas.dmtf.org/cimi/2">
    <id> xs:anyURI </id>
    <name> xs:string </name>
    <description> xs:string </description>
    <created> xs:dateTime </created>
    <updated> xs:dateTime </updated>
    <parent> xs:anyURI </parent>
    <properties>
        <property key="xs:string"> xs:string </property> *
    </properties>
    <vscope> valueScope </vscope> *
    <cpu> xs:integer </cpu>
    <memory> xs:integer </memory>
    <disks>
        <disk>
            <capacity> xs:integer </capacity>
            <format> xs:string </format>
            <initialLocation> xs:string </initialLocation>
        </disk> *
    </disks>
    <cpuArch> xs:string </cpuArch>
    <cpuSpeed> xs:integer </cpuSpeed>
    <operations>
        <operation rel="edit" href="xs:anyURI"/>
        <operation rel="delete" href="xs:anyURI"/>
    </operations>
</MachineConfiguration>
```

### 3.2.8 MachineConfigurationCollection serialization

The following pseudo-schemas (see notation in CIMI 1.3) describe the serialization of the Resource in both JSON and XML:

**JSON serialization:**

```json
{
    "resourceURI": 
        "http://schemas.dmtf.org/cimi/2/MachineConfigurationCollection",
    "id": string,
    "updated": string,
    "parent": string,
}
```
"count": number,
"machineConfigurations": [
    { "resourceURI": "http://schemas.dmtf.org/cimi/2/MachineConfiguration",
      "id": string,
      ... remaining MachineConfiguration attributes ...
    }, +
  ],
"operations": [ { "rel": "add", "href": string } ]
...}

**XML serialization:**

```xml
<Collection
    resourceURI="http://schemas.dmtf.org/cimi/2/MachineConfigurationCollection"
    xmlns="http://schemas.dmtf.org/cimi/2"
    id xs:anyURI /
    <updated> xs:dateTime </updated>
    <parent> xs:anyURI </parent>
    <count> xs:integer </count>
    <machineConfigurations>
        <MachineConfiguration>
            <id> xs:anyURI </id>
            ... remaining MachineConfiguration attributes ...
        </MachineConfiguration> *
    </machineConfigurations>
    <operations>
        <operation rel="add" href="xs:anyURI"/>
    </operations>
    <xs:any>*
</Collection>
```

### 3.2.9 MachineImage serialization

The following pseudo-schemas (see notation in CIMI 1.3) describe the serialization of the Resource in both JSON and XML:

**JSON media type:** application/json

**JSON serialization:**

```json
{
    "resourceURI": "http://schemas.dmtf.org/cimi/2/MachineImage",
    "id": string,
    "name": string,
    "description": string,
    "created": string,
```
"updated": string,
"parent": string,
"properties": [ string: string, + ],
"state": string,
"type": string,
"imageLocation": string,
"relatedImage": [ "href": string ],
"operations": [ { "rel": "edit", "href": string }, { "rel": "delete", "href": string } ]
...
}

**XML media type:** application/xml

**XML serialization:**

```xml
<MachineImage xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <name> xs:string </name>
  <description> xs:string </description>
  <created> xs:dateTime </created>
  <updated> xs:dateTime </updated>
  <parent> xs:anyURI </parent>
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties>
  <state> xs:string </state>
  <type> xs:string </type>
  <imageLocation> xs:anyURI </imageLocation>
  <relatedImage href="xs:anyURI"/>
  <operations>
    <operation rel="edit" href="xs:anyURI"/>
    <operation rel="delete" href="xs:anyURI"/>
  </operations>
  <xs:any>*
</MachineImage>
```

### 3.2.10 MachineImageCollection serialization

The following pseudo-schemas (see notation in CIMI 1.3) describe the serialization of the Resource in both JSON and XML:
JSON serialization:

```json
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/MachineImageCollection",
 "id": string,
 "updated": string,
 "parent": string,
 "count": number,
 "machineImages": [ 
 { "resourceURI": "http://schemas.dmtf.org/cimi/2/MachineImage",
  "id": string,
  ... remaining MachineImage attributes ... 
 }, +
],
 "operations": [ { "rel": "add", "href": string } ]
...
}
```

XML serialization:

```xml
<Collection resourceURI="http://schemas.dmtf.org/cimi/2/MachineImageCollection"
  xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <updated> xs:dateTime </updated>
  <parent> xs:anyURI </parent>
  <count> xs:integer </count>
  <machineImages>
    <MachineImage>
      <id> xs:anyURI </id>
      ... remaining MachineImage attributes ...
    </MachineImage> *
  </machineImages>
  <operations>
    <operation rel="add" href="xs:anyURI"/>
  </operations>
  <xs:any>*
</Collection>
```

### 3.3 Credential Resources group

#### 3.3.1 Credential serialization

The following pseudo-schemas (see notation in CIMI 1.3) describe the serialization of the Resource in both JSON and XML:
JSON media type: application/json

JSON serialization:
```json
{  "resourceURI": "http://schemas.dmtf.org/cimi/2/Credential",  "id": string,  "name": string,  "description": string,  "created": string,  "updated": string,  "parent": string,  "properties": { string: string, + },  "operations": [    { "rel": "edit", "href": string },    { "rel": "delete", "href": string }  ]}
```

XML media type: application/xml

XML serialization:
```xml
<Credential xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <name> xs:string </name>
  <description> xs:string </description>
  <created> xs:dateTime </created>
  <updated> xs:dateTime </updated>
  <parent> xs:anyURI </parent>
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties>
  <operations>
    <operation rel="edit" href="xs:anyURI"/>
    <operation rel="delete" href="xs:anyURI"/>
  </operations>
  <xs:any>*
</Credential>
```

3.3.2 CredentialCollection serialization

The following pseudo-schemas (see notation in CIMI 1.3) describe the serialization of the Resource in both JSON and XML:
JSON serialization:
```json
{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/CredentialCollection",
  "id": string,
  "updated": string,
  "parent": string,
  "count": number,
  "credentials": [
    {
      "resourceURI": "http://schemas.dmtf.org/cimi/2/Credential",
      "id": string,
      "credentials": [
        {
          "resourceURI": "http://schemas.dmtf.org/cimi/2/Credential",
          "id": string,
          ... remaining Credential attributes ...
        }
      ]
    }, ...
  ],
  "operations": [{ "rel": "add", "href": string } ]
}
```

XML serialization:
```xml
<Collection resourceURI="http://schemas.dmtf.org/cimi/2/CredentialCollection"
  xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <updated> xs:dateTime </updated>
  <parent> xs:anyURI </parent>
  <count> xs:integer </count>
  <credentials>
    <Credential>
      <id> xs:anyURI </id>
      ... remaining Credential attributes ...
    </Credential> *
  </credentials>
  <operations>
    <operation rel="add" href="xs:anyURI"/>
  </operations>
  <xs:any>*
</Collection>
```

### 3.3.3 CredentialTemplate serialization

The following pseudo-schemas (see notation in CIMI 1.3) describe the serialization of the Resource in both JSON and XML:
JSON media type: application/json

JSON serialization:

```json
{
"resourceURI": "http://schemas.dmtf.org/cimi/2/CredentialTemplate",
"id": string,
"name": string, ?
"description": string, ?
"created": string, ?
"updated": string, ?
"parent": string, ?
"properties": { string: string, + }, ?
"operations": [
    { "rel": "edit", "href": string }, ?
    { "rel": "delete", "href": string } ?
] ?
...}
```

XML media type: application/xml

XML serialization:

```xml
<CredentialTemplate xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <name> xs:string </name> ?
  <description> xs:string </description> ?
  <created> xs:dateTime </created> ?
  <updated> xs:dateTime </updated> ?
  <parent> xs:anyURI </parent> ?
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties>
  <operations>
    <operation rel="edit" href="xs:anyURI"/> ?
    <operation rel="delete" href="xs:anyURI"/> ?
  </operations>
  <xs:any>*
</CredentialTemplate>
```

3.3.4 CredentialTemplateCollection serialization

The following pseudo-schemas (see notation in CIMI 1.3) describe the serialization of the Resource in both JSON and XML:
JSON serialization:

```json
{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/CredentialTemplateCollection",
  "id": string,
  "updated": string,
  "parent": string,
  "count": number,
  "credentialTemplates": [
    {
      "resourceURI": "http://schemas.dmtf.org/cimi/2/CredentialTemplate",
      "id": string,
      ... remaining CredentialTemplate attributes ...
    }, +
  ],
  "operations": [ { "rel": "add", "href": string } ]
}
```

XML serialization:

```xml
<Collection
  resourceURI="http://schemas.dmtf.org/cimi/2/CredentialTemplateCollection"
  xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <updated> xs:dateTime </updated>
  <parent> xs:anyURI </parent>
  <count> xs:integer </count>
  <credentialTemplates>
    <CredentialTemplate>
      <id> xs:anyURI </id>
      ... remaining CredentialTemplate attributes ...
    </CredentialTemplate> *
  </credentialTemplates>
  <operation rel="add" href="xs:anyURI"/> ?
  <xs:any>*
</Collection>
```
3.4 Volume Resources group

3.4.1 Volume Resources group and their relationships

The group of Volume-related Resources is represented in Figure 7, with other associated Resources.

![Volume Resources group diagram](image)

Figure 7 – Volume Resources group

3.4.2 Volume serialization

The following pseudo-schemas (see notation in CIMI 1.3) describe the serialization of the Resource in both JSON and XML:

**JSON media type**: application/json

**JSON serialization**:

```json
{
    "resourceURI": "http://schemas.dmtf.org/cimi/2/Volume",
    "id": string,
    "name": string, ?
    "description": string, ?
    "created": string, ?
    "updated": string, ?
    "parent": string, ?
    "properties": [ string: string, + ], ?
```
"state": string,
"type": string,
"capacity": number,
"bootable": boolean,
"images": { "href": string }, ?
"meters": { "href": string }, ?
"eventLog": { "href": string }, ?
"operations": [ 
  { "rel": "edit", "href": string }, ?
  { "rel": "delete", "href": string } ?
] ?
...

XML media type: application/xml

XML serialization:

```xml
<Volume xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <name> xs:string </name> ?
  <description> xs:string </description> ?
  <created> xs:dateTime </created> ?
  <updated> xs:dateTime </updated> ?
  <parent> xs:anyURI </parent> ?
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties>
  <state> xs:string </state>
  <type> xs:anyURI </type>
  <capacity> xs:integer </capacity>
  <bootable> xs:boolean </bootable>
  <images href="xs:anyURI"/> ?
  <meters href="xs:anyURI"/> ?
  <eventLog href="xs:anyURI"/> ?
  <operations>
    <operation rel="edit" href="xs:anyURI"/> ?
    <operation rel="delete" href="xs:anyURI"/> ?
  </operations>
  <xs:any>*
</Volume>
```
3.4.3 VolumeCollection serialization

The following pseudo-schemas (see notation in CIMI 1.3) describe the serialization of the Resource in both JSON and XML:

**JSON serialization:**

```json
{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/VolumeCollection",
  "id": "string",
  "updated": "string",
  "parent": "string",
  "count": "number",
  "volumes": [
    {
      "resourceURI": "http://schemas.dmtf.org/cimi/2/Volume",
      "id": "string",
      ... remaining Volume attributes ...
    }, +
    ?
  ],
  "operations": [
    { "rel": "add", "href": "string" } ?
  ]
}
```

**XML serialization:**

```xml
<Collection resourceURI="http://schemas.dmtf.org/cimi/2/VolumeCollection"
            xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <updated> xs:dateTime </updated>
  <parent> xs:anyURI </parent>
  <count> xs:integer </count>
  <volumes>
    <Volume>
      <id> xs:anyURI </id>
      ... remaining Volume attributes ...
    </Volume> *
  </volumes>
  <operations>
    <operation rel="add" href="xs:anyURI"/> ?
  </operations>
  <xs:any>*
</Collection>
```
3.4.4 locatedVolumeCollection serialization

This collection is using an additional accessory attribute (the initial location of the Volume), compared with the basic Volume collection.

The following pseudo-schemas (see notation in CIMI 1.3) describe the serialization of the Resource in both JSON and XML:

**JSON serialization:**

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/locatedVolumeCollection",
 "id": string,
 "updated": string,
 "parent": string,
 "count": number,
 "locatedVolumes": [
   { "resourceURI": "http://schemas.dmtf.org/cimi/2/locatedVolume",
     "id": string,
     "name": string, ?
     "description": string, ?
     "created": string, ?
     "updated": string, ?
     "parent": string, ?
     "properties": { string: string, + }, ?
     "initialLocation": string, ?
     "volume": { "href": string },
     "operations": [
       { "rel": "edit", "href": string }, ?
       { "rel": "delete", "href": string } ?
     ] ?
   ...,
   ], +
 }, ?

"operations": [  
  { "rel": "add", "href": string } ?
  { "rel": "insert", "href": string } ?
  { "rel": "remove", "href": string } ?
 ]
 ...
}
```

**XML serialization:**

```
<Collection
   resourceURI="http://schemas.dmtf.org/cimi/2/locatedVolumeCollection"
```
3.4.5 VolumeTemplate serialization

The following pseudo-schemas (see notation in CIMI 1.3) describe the serialization of the Resource in both JSON and XML:

**JSON media type**: application/json

**JSON serialization**:

```json
{
    "resourceURI": "http://schemas.dmtf.org/cimi/2/VolumeTemplate",
    "id": string,
    "locatedVolumes": [  
        {
            "id": string,
            "name": string,
            "description": string,
            "created": dateTime,
            "updated": dateTime,
            "parent": string,
            "properties": {  
                "property": string,
            },
            "initialLocation": string,
            "volume": string,
            "operations": [  
                {"rel" : "edit", "href": string},
                {"rel" : "delete", "href": string}
            ]
        },
      ]
}
```
Cloud Infrastructure Management Interface (CIMI) Primer

"name": string, ?
"description": string, ?
"created": string, ?
"updated": string, ?
"parent": string, ?
"properties": [ string: string, + ], ?
"volumeConfig": {
  "href": string | ... VolumeConfiguration attributes ...
},
"volumeImage": { "href": string }, ?
"meterTemplates": [
  { "href": string, ?
    ... MeterTemplate attributes ... ?
  }, *
], ?
"eventLogTemplate": [
  "href": string, ?
  ... EventLogTemplate attributes ... ?
], ?,
"genResourceMetadata": { "href": string } ?
"operations": [
  { "rel": "edit", "href": string }, ?
  { "rel": "delete", "href": string } ?
] ?
...
3.4.6 VolumeTemplateCollection serialization

The following pseudo-schemas (see notation in CIMI 1.3) describe the serialization of the Resource in both JSON and XML:

This Resource shall be serialized as follows:

**JSON serialization:**

```
{
    "resourceURI": "http://schemas.dmtf.org/cimi/2/VolumeTemplateCollection",
    "id": string,
    "updated": string,
    "parent": string,
    "count": number,
    "volumeTemplates": [
        {
            "resourceURI": "http://schemas.dmtf.org/cimi/2/VolumeTemplate",
            "id": string,
            "... remaining volumeTemplate attributes ...
        }, +
    ], ?
    "operations": [ [ "rel": "add", "href": string ] ? ]
    ...
}
```

XML serialization:

```xml
<Collection
    resourceURI="http://schemas.dmtf.org/cimi/2/VolumeTemplateCollection"
    xmlns="http://schemas.dmtf.org/cimi/2">
    <id> xs:anyURI </id>
    <updated> xs:dateTime </updated>
    <parent> xs:anyURI </parent>
    <count> xs:integer </count>
    <volumeTemplates>
        <VolumeTemplate>
            <id> xs:anyURI </id>
            ... remaining VolumeTemplates attributes ...
        </VolumeTemplate> *
    </volumeTemplates>
    <operations>
        <operation rel="add" href="xs:anyURI"/>
        <operation rel="delete" href="xs:anyURI"/>
    </operations>
    <xs:any>*
</Collection>
```

3.4.7 VolumeConfiguration serialization

The following pseudo-schemas (see notation in CIMI 1.3) describe the serialization of the Resource in both JSON and XML:

- **JSON media type**: application/json

JSON serialization:

```json
{
    "resourceURI": "http://schemas.dmtf.org/cimi/2/VolumeConfiguration",
    "id": string,
    "name": string, ?
    "description": string, ?
    "created": string, ?
    "updated": string, ?
    "parent": string, ?
    "properties": { string: string, + }, ?,
    "type": string,
    "format": string,
    "capacity": number,
    "operations": [
        { "rel": "edit", "href": string }, ?
        { "rel": "delete", "href": string } ?
    ] ?
```
XML media type: application/xml

XML serialization:

```xml
<VolumeConfiguration xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <name> xs:string </name> ?
  <description> xs:string </description> ?
  <created> xs:dateTime </created> ?
  <updated> xs:dateTime </updated> ?
  <parent> xs:anyURI </parent> ?
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties> ?
  <type> xs:anyURI </type>
  <format> xs:string </format>
  <capacity> xs:integer </capacity>
  <operations>
    <operation rel="edit" href="xs:anyURI"/> ?
    <operation rel="delete" href="xs:anyURI"/> ?
  </operations>
  <xs:any>*
</VolumeConfiguration>
```

3.4.8 VolumeConfigurationCollection serialization

The following pseudo-schemas (see notation in CIMI 1.3) describe the serialization of the Resource in both JSON and XML:

**JSON serialization:*

```json
{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/VolumeConfigurationCollection",
  "id": string,
  "updated": string,
  "parent": string,
  "count": number,
  "volumeConfigurations": [
    {
      "resourceURI": "http://schemas.dmtf.org/cimi/2/VolumeConfiguration",
      "id": string,
      ... remaining VolumeConfiguration attributes ...
    }, +
  ], ?
}
"operations": [ { "rel": "add", "href": string } ? ]
...}

XML serialization:

```xml
<Collection
    resourceURI="http://schemas.dmtf.org/cimi/2/VolumeConfigurationCollection"
    xmlns="http://schemas.dmtf.org/cimi/2">
    <id> xs:anyURI </id>
    <updated> xs:dateTime </updated>
    <parent> xs:anyURI </parent>
    <count> xs:integer </count>
    <volumeConfigurations>
        <VolumeConfiguration>
            <id> xs:anyURI </id>
            ... remaining VolumeConfiguration attributes ...
        </VolumeConfiguration> *
    </volumeConfigurations>
    <operations>
        <operation rel="add" href="xs:anyURI"/> ?
    </operations>
    <xs:any>*
</Collection>
```

3.4.9 VolumeImage serialization

The following pseudo-schemas (see notation in CIMI 1.3) describe the serialization of the Resource in both JSON and XML:

**JSON media type:** application/json

**JSON serialization:**

```json
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/VolumeImage",
  "id": string,
  "name": string, ?
  "description": string, ?
  "created": string, ?
  "updated": string, ?
  "parent": string, ?
  "properties": { string: string, + }, ?
  "state": string,
  "imageLocation": string,
  "bootable": boolean,
  "operations": [
XML media type: application/xml

XML serialization:

```xml
<VolumeImage xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <name> xs:string </name> ?
  <description> xs:string </description> ?
  <created> xs:dateTime </created> ?
  <updated> xs:dateTime </updated> ?
  <parent> xs:anyURI </parent> ?
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties> ?
  <state> xs:string </state>
  <imageLocation>xs:anyURI</imageLocation>
  <bootable> xs:boolean </bootable>
  <operations>
    <operation rel="edit" href="xs:anyURI"/> ?
    <operation rel="delete" href="xs:anyURI"/> ?
  </operations>
  <xs:any>*
</VolumeImage>
```

3.4.10 VolumeImageCollection serialization

The following pseudo-schemas (see notation in CIMI 1.3 describe the serialization of the Resource in both JSON and XML:

JSON serialization:

```json
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/VolumeImageCollection",
  "id": string,
  "updated": string,
  "parent": string,
  "count": number,
  "volumeImages": [
    { "resourceURI": "http://schemas.dmtf.org/cimi/2/VolumeImage",
      "id": string,
      ... remaining VolumeImage attributes ...
  ]
}
"operations": [ { "rel": "add", "href": string } ? ]
...
}

XML serialization:

```xml
<Collection resourceURI="http://schemas.dmtf.org/cimi/2/VolumeImageCollection"
    xmlns="http://schemas.dmtf.org/cimi/2">
  <id>xs:anyURI</id>
  <updated>xs:dateTime</updated>
  <parent>xs:anyURI</parent>
  <count>xs:integer</count>
  <volumeImages>
    <VolumeImage>
      <id>xs:anyURI</id>
      ... remaining VolumeImage attributes ...
    </VolumeImage> *
  </volumeImages>
  <operations>
    <operation rel="add" href="xs:anyURI"/> ?
  </operations>
  <xs:any>*
</Collection>
```

### 3.5 System Resources group

#### 3.5.1 System Resources and their relationships

The group of System-related Resources is represented in Figure 8, with other associated Resources related to System:
System Resources Group (1)

Resources related to SystemTemplate are shown in Figure 9:
3.5.2 System serialization

The following pseudo-schemas (see notation in CIMI 1.3) describe the serialization of the Resource in both JSON and XML:

**JSON media type: application/json**

**JSON serialization:**

```json
{
    "resourceURI": "http://schemas.dmtf.org/cimi/2/System",
    "id": string,
    "name": string, ?
    "description": string, ?
    "created": string, ?
    "updated": string, ?
    "parent": string, ?
    "properties": { string: string, + }, ?
    "state": string,
    "systems": { "href": string }, ?
    "machines": { "href": string }, ?
    "credentials": { "href": string }, ?
}
```
"volumes": { "href": string }, ?
"networks": { "href": string }, ?
"networkServices": { "href": string }, ?
"meters": { "href": string }, ?
"eventLog": { "href": string }, ?
"operations": [
  { "rel": "edit", "href": string, ("available": boolean)? }, ?,
  { "rel": "delete", "href": string, ("available": boolean)? }, ?,
  { "rel": "http://schemas.dmtf.org/cimi/2/action/start", "href": string,
    ("available": boolean)? }, ?,
  { "rel": "http://schemas.dmtf.org/cimi/2/action/stop", "href": string,
    ("available": boolean)? }, ?,
  { "rel": "http://schemas.dmtf.org/cimi/2/action/restart", "href": string,
    ("available": boolean)? }, ?,
  { "rel": "http://schemas.dmtf.org/cimi/2/action/pause", "href": string,
    ("available": boolean)? }, ?,
  { "rel": "http://schemas.dmtf.org/cimi/2/action/suspend", "href": string,
    ("available": boolean)? }, ?,
  { "rel": "http://schemas.dmtf.org/cimi/2/action/export", "href": string,
    ("available": boolean)? }, ?
] ?
...

XML media type: application/xml
XML serialization:

```xml
<System xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <name> xs:string </name> ?
  <description> xs:string </description> ?
  <created> xs:dateTime </created> ?
  <updated> xs:dateTime </updated> ?
  <parent> xs:anyURI </parent> ?
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties>
  <state> xs:string </state>
  <systems href="xs:anyURI"/> ?
  <machines href="xs:anyURI"/> ?
  <credentials href="xs:anyURI"/> ?
  <volumes href="xs:anyURI"/> ?
  <networks href="xs:anyURI"/> ?
  <networkServices href="xs:anyURI"/> ?
</System>
```
3.5.3 SystemCollection serialization

The following pseudo-schemas (see notation in CIMI 1.3) describe the serialization of the Resource in both JSON and XML:

**JSON serialization:**

```json
{
    "resourceURI": "http://schemas.dmtf.org/cimi/2/SystemCollection",
    "id": string,
    "updated": string,
    "parent": string,
    "count": number,
    "systems": [
        {
            "resourceURI": "http://schemas.dmtf.org/cimi/2/System",
            "id": string,
            ... remaining System attributes ...
        }, +
    ],
    "operations": [
        { "rel": "add", "href": string }, ?
        { "rel": "remove", "href": string } ?
    ]
}
```
3.5.4 **SystemService serialization**

The following pseudo-schemas (see notation in CIMI 1.3) describe the serialization of the Resource in both JSON and XML:

**JSON media type**: application/json

**JSON serialization**:

```json
{
    "resourceURI": "http://schemas.dmtf.org/cimi/2/SystemService",
    "id": string,
    "name": string, ?
    "description": string, ?
    "created": string, ?
    "updated": string, ?
    "parent": string, ?
}
```

**XML serialization**:

```xml
<Collection resourceURI="http://schemas.dmtf.org/cimi/2/SystemCollection"
    xmlns="http://schemas.dmtf.org/cimi/2">
    <id> xs:anyURI </id>
    <updated> xs:dateTime </updated>
    <parent> xs:anyURI </parent>
    <count> xs:integer </count>
    <systems>
        <System>
            <id> xs:anyURI </id>
            ... remaining System attributes ...
        </System> *
    </systems>
    <operations>
        <operation rel="add" href="xs:anyURI"/> ?
        <operation rel="remove" href="xs:anyURI"/> ?
        <operation rel="insert" href="xs:anyURI"/> ?
        <operation rel="http://schemas.dmtf.org/cimi/2/action/import" href="xs:anyURI"/> ?
    </operations>
</Collection>
```
"properties": { string: string, + }, ?
"serviceType": string,
"machines": { "href": string }, ?
"volumes": { "href": string }, ?
"systems": { "href": string }, ?

.."parameters": { string: string, + }, ?

"meters": { "href": string }, ?
"eventLog": { "href": string }, ?
"operations": [ ]?

XML media type: application/xml

XML serialization:

```xml
<SystemService xmlns="http://schemas.dmtf.org/cimi/2">
  <id xs:anyURI /></id>
  <name xs:string /></name> ?
  <description xs:string /></description> ?
  <created xs:dateTime /></created> ?
  <updated xs:dateTime /></updated> ?
  <parent xs:anyURI /></parent> ?
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties> ?
  <state xs:string /></state>
  <serviceType xs:string /></type>
  <machines href="xs:anyURI"/> *
  <volumes href="xs:anyURI"/> *
  <systems href="xs:anyURI"/> *
  <parameters key="xs:string"> xs:string </parameters> *

  <meters href="xs:anyURI"/> ?
  <eventLog" href="xs:anyURI"/> ?
  <operations>
```
3.5.5 SystemTemplate serialization

The following pseudo-schemas (see notation in CIMI 1.3) describe the serialization of the Resource in both JSON and XML:

**JSON media type:** application/json

**JSON serialization:**

```json
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/SystemTemplate",
  "id": string,
  "name": string, 
  "description": string, 
  "created": string, 
  "updated": string, 
  "parent": string, 
  "properties": { string: string, + }, 
  "componentDescriptors": [ 
    { "name": string, 
      "description": string, 
      "properties": { string: string, + }, 
      "type": string, 
      "componentResource": { 
        "href": string, 
        ... ComponentResource attributes ... 
      },
      "quantity": number 
    }, +
  ], 
  "serviceDescriptors": [ 
    { "name": string, 
      "description": string, 
      "properties": { string: string, + }, 
      "serviceType": string, 
      ... 
    }, +
  ], 
  "meterTemplates": [ 
```
XML media type: application/xml

XML serialization:

```
<SystemTemplate xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <name> xs:string </name> ?
  <description> xs:string </description> ?
  <created> xs:dateTime </created> ?
  <updated> xs:dateTime </updated> ?
  <parent> xs:anyURI </parent> ?
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties>
  <componentDescriptors>
    <componentDescriptor>
      <name> xs:string </name> ?
      <description> xs:string </description> ?
      <properties>
        <property key="xs:string"> xs:string </property> *
      </properties> ?
    </componentDescriptor>
    <type> xs:anyURI </type>
    <componentResource href="xs:anyURI"? >
```

```
... ComponentResource attributes ...

</componentResource>

<quantity> xs:integer </quantity>
</componentDescriptor> *
</componentDescriptors>

<serviceDescriptors>
<serviceDescriptor>
<name> xs:string </name> ?
<description> xs:string </description> ?
<properties>
  <property key="xs:string"> xs:string </property> *
</properties> ?
<serviceType> xs:anyURI </serviceType>
...
</serviceDescriptor> *
</serviceDescriptors>

<meterTemplates>
<meterTemplate href="xs:anyURI"? >
... MeterTemplate attributes ...
</meterTemplate> *
</meterTemplates>
<eventLogTemplate href="xs:anyURI"? >
... EventLogTemplate attributes ...
</eventLogTemplate> ?
<importImage > xs:anyURI </importImage> ?
<genResourceMetadata href="xs:anyURI"? /> ?
<operations>
<operation rel="edit" href="xs:anyURI"/> ?
<operation rel="delete" href="xs:anyURI"/> ?
<operation rel="http://schemas.dmtf.org/cimi/2/action/export" href="xs:anyURI"/> ?
</operations>
</any>
</SystemTemplate>

3.5.6 SystemTemplateCollection serialization

The following pseudo-schemas (see notation in CIMI 1.3) describe the serialization of the Resource in both JSON and XML:
This Resource shall be serialized as follows:

**JSON serialization:**

```json
{
   "resourceURI": "http://schemas.dmtf.org/cimi/2/SystemTemplateCollection",
   "id": string,
   "updated": string,
   "parent": string,
   "count": number,
   "systemTemplates": [
      { "resourceURI": "http://schemas.dmtf.org/cimi/2/SystemTemplate",
        "id": string,
        ... remaining SystemTemplate attributes ...
      }, +
   ],
   "operations": [
      { "rel": "add", "href": string }, ?
      { "rel": "http://schemas.dmtf.org/cimi/2/action/import", "href": string } ?
   ]
}
```

**XML serialization:**

```xml
<Collection
   resourceURI="http://schemas.dmtf.org/cimi/2/SystemTemplateCollection"
   xmlns="http://schemas.dmtf.org/cimi/2">
   <id> xs:anyURI </id>
   <updated> xs:dateTime </updated>
   <parent> xs:anyURI </parent>
   <count> xs:integer </count>
   <systemTemplates>
      <SystemTemplate>
         <id> xs:anyURI </id>
         ... remaining SystemTemplate attributes ...
      </SystemTemplate> *
   </systemTemplates>
   <operations>
      <operation rel="add" href="xs:anyURI"/> ?
      <operation rel="http://schemas.dmtf.org/cimi/2/action/import"
                 href="xs:anyURI"/> ?
   </operations>
   <xs:any>*
</Collection>
```
3.6 Network Resources group

3.6.1 Network Resources and their relationships

The group of Network-related Resources and their relationships is represented in Figure 10:

Figure 10 – Network Resources group

How network segments and endpoint resources relate to other resources, including Machines is shown in Figure 11. In particular, NetworkService resources may affect only parts of a Network (i.e., some ProtocolSegments). A NetworkService resource may also affect more than one Network, depending on the nature of the service.
Network Resources group (2)
(Resource Structure Relationships)

Figure 11 – Network Resources group: Resource structure relationships

Figure 12 illustrates more particularly how Machine resources connect to Network resources.
Network Resources group (3)
(Network Connectivity Perspective)

Figure 12 – Network Resources group: Network connectivity perspective

3.6.2 Network serialization

The following pseudo-schemas (see notation in CIMI 1.3) describe the serialization of the Resource in both JSON and XML:

**JSON media type:** application/json

**JSON serialization:**

```json
{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/Network",
  "id": string,
  "name": string, ?
  "description": string, ?
  "created": string, ?
  "updated": string, ?
  "parent": string, ?
  "properties": { string: string, + }, ?
  "state": string,
  "segments": { "href": string },
  "services": { "href": string },
  "subnetworks": { "href": string }, ?
}
```
"meters": [ "href": string ], ?
"eventLog": { "href": string }, ?
"operations": [
  { "rel": "edit", "href": string }, ?
  { "rel": "delete", "href": string }, ?
  { "rel": "http://schemas.dmtf.org/cimi/2/action/start", "href": string }, ?
  { "rel": "http://schemas.dmtf.org/cimi/2/action/stop", "href": string } ?
] ?
...

XML media type: application/xml

XML serialization:

```xml
<Network xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <name> xs:string </name> ?
  <description> xs:string </description> ?
  <created> xs:dateTime </created> ?
  <updated> xs:dateTime </updated> ?
  <parent> xs:anyURI </parent> ?
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties> ?
  <state> xs:string </state>
  <segments href="xs:anyURI"/>
  <services href="xs:anyURI"/>
  <subnetworks href="xs:anyURI"/> ?
  <meters href="xs:anyURI"/> ?
  <eventLog" href="xs:anyURI"/> ?
  <operation rel="edit" href="xs:anyURI"/> ?
  <operation rel="delete" href="xs:anyURI"/> ?
  <operations>
    <operation rel="http://schemas.dmtf.org/cimi/2/action/start"
      href="xs:anyURI"/> ?
    <operation rel="http://schemas.dmtf.org/cimi/2/action/stop"
      href="xs:anyURI"/> ?
  </operations>
  <xs:any> *
</Network>
```
3.6.3 NetworkCollection serialization

The following pseudo-schemas (see notation in CIMI 1.3) describe the serialization of the Resource in both JSON and XML:

This Resource shall be serialized as follows:

**JSON serialization:**

```json
{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/NetworkCollection",
  "id": string,
  "updated": string,
  "parent": string,
  "count": number,
  "networks": [
    {
      "resourceURI": "http://schemas.dmtf.org/cimi/2/Network",
      "id": string,
      ... remaining Network attributes ...
    }, +
  ],
  "operations": [ { "rel": "add", "href": string } ? ]
}
```

**XML serialization:**

```xml
<Collection resourceURI="http://schemas.dmtf.org/cimi/2/NetworkCollection"
 xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <updated> xs:dateTime </updated>
  <parent> xs:anyURI </parent>
  <count> xs:integer </count>
  <networks>
    <Network>
      <id> xs:anyURI </id>
      ... remaining Network attributes ...
    </Network> *
  </networks>
  <operations>
    <operation rel="add" href="xs:anyURI"/> ?
  </operations>
  <xs:any>*
</Collection>
```
3.6.4 NetworkTemplate serialization

The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML:

**JSON media type:** application/json

**JSON serialization:**

```json
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/NetworkTemplate",
  "id": string,
  "name": string, ?
  "description": string, ?
  "created": string, ?
  "updated": string, ?
  "parent": string, ?
  "properties": { string: string, + }, ?,
  "initialState": string, ?
  "segments": { "href": string }, ?,
  "segmentTemplates": [
    { "href": string, ?
      ... ProtocolSegmentTemplate attributes ... ?
    }, *
  ], ?,
  "services": { "href": string }, ?,
  "serviceTemplates": [
    { "href": string, ?
      ... NetworkServiceTemplate attributes ... ?
    }, *
  ], ?,
  "subnetworks": { "href": string }, ?,
  "subnetworkTemplates": [
    { "href": string, ?
      ... NetworkTemplate attributes ... ?
    }, *
  ], ?,
  "meterTemplates": [
    { "href": string, ?
      ... MeterTemplate attributes ... ?
    }, *
  ], ?,
  "eventLogTemplate": { "href": string, ?
```
... EventLogTemplate attributes ... ?

"operations": [
    { "rel": "edit", "href": string }, ?
    { "rel": "delete", "href": string } ?
] ?
...

XML media type: application/xml

XML serialization:

```xml
<NetworkTemplate xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <name> xs:string </name> ?
  <description> xs:string </description> ?
  <created> xs:dateTime </created> ?
  <updated> xs:dateTime </updated> ?
  <parent> xs:anyURI </parent> ?
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties> ?
  <initialState> xs:string </initialState> ?
  <segments href="xs:anyURI"/> ?
  <segmentTemplates>
    <SegmentTemplate href="xs:anyURI"/>
    ... ProtocolSegmentTemplate attributes ... ?
    </SegmentTemplate> *
  </segmentTemplates>
  <services href="xs:anyURI"/> ?
  <serviceTemplates>
    <ServiceTemplate href="xs:anyURI"/>
    ... NetworkServiceTemplates attributes ... ?
    </ServiceTemplate> *
  </serviceTemplates>
  <subnetworks href="xs:anyURI"/>
  <subnetworkTemplates>
    <NetworkTemplate href="xs:anyURI"/>
    ... NetworkTemplate attributes ... ?
    </NetworkTemplate> *
  </subnetworkTemplates>
</NetworkTemplate>
```
3.6.5 NetworkTemplateCollection serialization

The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML:

This Resource shall be serialized as follows:

**JSON serialization:**

```json
dict = {
    "resourceURI": "http://schemas.dmtf.org/cimi/2/NetworkTemplateCollection",
    "id": string,
    "updated": string,
    "parent": string,
    "count": number,
    "networkTemplates": [
        {
            "resourceURI": "http://schemas.dmtf.org/cimi/2/NetworkTemplate",
            "id": string,
            ... remaining NetworkTemplate attributes ...
        },
        ...
    ],
    "operations": [
        { "rel": "add", "href": string } ?
        ...
    ]
}
```

**XML serialization:**

```xml
<Collection
    resourceURI="http://schemas.dmtf.org/cimi/2/NetworkTemplateCollection"
    xmlns="http://schemas.dmtf.org/cimi/2">
    <id> xs:anyURI </id>
    <updated> xs:dateTime </updated>
```
3.6.6 NetworkService serialization

The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML.

**JSON media type: application/json**

**JSON serialization:**

```json
{
    "resourceURI": "http://schemas.dmtf.org/cimi/2/NetworkService",
    "id": string,
    "name": string, ?
    "description": string, ?
    "created": string, ?
    "updated": string, ?
    "parent": string, ?
    "properties": { string: string, + }, ?
    "state": string,
    "type": string,
    "endpoints": { "href": string }, ?,
    "segments": { "href": string }, ?,
    .."policies": { string: string, + }, ?
    "meters": { "href": string }, ?
    "eventLog": { "href": string }, ?
    "operations": [
        { "rel": "edit", "href": string }, ?,
        { "rel": "delete", "href": string }, ?,
        { "rel": "http://schemas.dmtf.org/cimi/2/action/start", "href": string }, ?
    ]
}
```
XML media type: application/xml

XML serialization:

```
<NetworkService xmlns="http://schemas.dmtf.org/cimi/2">
  <id>xs:anyURI</id>
  <name>xs:string</name> ?
  <description>xs:string</description> ?
  <created>xs:dateTime</created> ?
  <updated>xs:dateTime</updated> ?
  <parent>xs:anyURI</parent> ?
  <properties>
    <property key="xs:string">xs:string</property> *
  </properties> ?
  <state>xs:string</state>
  <type>xs:string</type>
  <endpoints href="xs:anyURI"/> *
  <segments href="xs:anyURI"/> *
  <policies key="xs:string">xs:string</policies> *
  <meters href="xs:anyURI"/> ?
  <eventLog" href="xs:anyURI"/> ?
  <operations>
    <operation rel="edit" href="xs:anyURI"/> ?
    <operation rel="delete" href="xs:anyURI"/> ?
    <operation rel="http://schemas.dmtf.org/cimi/2/action/start" href="xs:anyURI"/> ?
    <operation rel="http://schemas.dmtf.org/cimi/2/action/stop" href="xs:anyURI"/> ?
  </operations>
  <xs:any/>
</NetworkService>
```

### 3.6.7 NetworkServiceCollection serialization

The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML.

This Resource shall be serialized as follows:
JSON serialization:

```json
{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/NetworkServiceCollection",
  "id": string,
  "updated": string,
  "parent": string,
  "count": number,
  "services": [
    {
      "resourceURI": "http://schemas.dmtf.org/cimi/2/NetworkService",
      "id": string,
      … remaining NetworkService attributes …
    }, +
  ],
  "operations": [ { "rel": "add", "href": string } ]
}
```

XML serialization:

```xml
<Collection
  resourceURI="http://schemas.dmtf.org/cimi/2/NetworkServiceCollection"
  xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <updated> xs:dateTime </updated>
  <parent> xs:anyURI </parent>
  <count> xs:integer </count>
  <services>
    <NetworkService>
      <id> xs:anyURI </id>
      … remaining NetworkService attributes …
    </NetworkService> *
  </services>
  <operations>
    <operation rel="add" href="xs:anyURI"/> ?
  </operations>
  <xs:any>*
</Collection>
```
3.6.8 NetworkServiceTemplate serialization

The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML.

**JSON media type:** application/json

**JSON serialization:**

```json
{
"resourceURI": "http://schemas.dmtf.org/cimi/2/NetworkServiceTemplate",
"id": string,
"name": string, ?
"description": string, ?
"created": string, ?
"updated": string, ?
"parent": string, ?
"properties": { string: string, + }, ?
"network": { "href": string }, ?,
"initialState": string, ?
"type": string,
"endpoints": { "href": string }, *
"segments": { "href": string }, *

.."policies": { string: string, + }, ?

"meterTemplates": [
    { "href": string, ?
        ...
        MeterTemplate attributes...
    }, *
], ?,

"eventLogTemplate": {
    "href": string, ?
    ...
    EventLogTemplate attributes...
}, ?,

"operations": [
    { "rel": "edit", "href": string }, ?,
    { "rel": "delete", "href": string }?,
] ?
...
}
```

**XML media type:** application/xml

**XML serialization:**

```xml
<NetworkServiceTemplate xmlns="http://schemas.dmtf.org/cimi/2">
```

```xml
...
```
<id> xs:anyURI </id>
<name> xs:string </name> ?
<description> xs:string </description> ?
<created> xs:dateTime </created> ?
<updated> xs:dateTime </updated> ?
<parent> xs:anyURI </parent> ?
<properties>
   <property key="xs:string"> xs:string </property> *
</properties> ?
<network href="xs:anyURI"/> ?
<initialState> xs:string </initialState> ?
?type> xs:string </type>
<endpoints>
   <ProtocolEndpoint href="xs:anyURI"/> *
</endpoints>
<segments>
   <ProtocolSegment href="xs:anyURI"/> *
</segments>
<policies>
   <policy key="xs:string"> xs:string </policy> *
</policies> ?
<meterTemplates>
   <MeterTemplate href="xs:anyURI"?>
      ... MeterTemplate attributes ... ?
   </MeterTemplate> *
</meterTemplates>
<eventLogTemplate href="xs:anyURI"? >
   ... EventLogTemplate attributes ... ?
</eventLogTemplate> ?
<operations>
   <operation rel="edit" href="xs:anyURI"/> ?
   <operation rel="delete" href="xs:anyURI"/> ?
</operations>
<xs:any>*
</NetworkServiceTemplate>
3.6.9 NetworkServiceTemplateCollection serialization

The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML.

This Resource shall be serialized as follows:

**JSON serialization:**

```json
{  "resourceURI": "http://schemas.dmtf.org/cimi/2/NetworkServiceTemplateCollection",  "id": string,  "updated": string,  "parent": string,  "count": number,  "networkServiceTemplates": [    {      "resourceURI": "http://schemas.dmtf.org/cimi/2/NetworkServiceTemplate",      "id": string,      ... remaining NetworkServiceTemplate attributes ...    }, +  ],  "operations": [    {      "rel": "add",      "href": string    } ?  ]}
```

**XML serialization:**

```xml
<Collection  
  resourceURI="http://schemas.dmtf.org/cimi/2/NetworkServiceTemplateCollection"  
  xmlns="http://schemas.dmtf.org/cimi/2">  
  <id> xs:anyURI </id>  
  <updated> xs:dateTime </updated>  
  <parent> xs:anyURI </parent>  
  <count> xs:integer </count>  
  <networkServiceTemplates>  
    <NetworkServiceTemplate>  
      <id> xs:anyURI </id>  
      ... remaining NetworkServiceTemplate attributes ...  
    </NetworkServiceTemplate> *  
  </networkServiceTemplates>  
  <operations>  
    <operation rel="add" href="xs:anyURI"/> ?  
  </operations>  
  <xs:any>*  
</Collection>
```
3.6.10 ProtocolSegment serialization

The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML.

**JSON media type:** application/json

**JSON serialization:**

```json
{
    "resourceURI": "http://schemas.dmtf.org/cimi/2/ProtocolSegment",
    "id": string,
    "name": string, ?
    "description": string, ?
    "created": string, ?
    "updated": string, ?
    "parent": string, ?
    "properties": { string: string, + }, ?
    "state": string,
    "protocol": string,
    "noDefaultRouting": boolean,
    "endpoints": { "href": string },
    "parameters": { string: string, + }, ?
    "meters": { "href": string }, ?
    "eventLog": { "href": string }, ?
    "operations": [
        { "rel": "edit", "href": string }, ?
        { "rel": "delete", "href": string }, ?,
        { "rel": "http://schemas.dmtf.org/cimi/2/action/start", "href": string }, ?,
        { "rel": "http://schemas.dmtf.org/cimi/2/action/stop", "href": string } ?
    ] ?
    ...}
```

**XML media type:** application/xml

**XML serialization:**

```xml
<ProtocolSegment xmlns="http://schemas.dmtf.org/cimi/2">
    <id> xs:anyURI </id>
    <name> xs:string </name> ?
    <description> xs:string </description> ?
    <created> xs:dateTime </created> ?
    <updated> xs:dateTime </updated> ?
    <parent> xs:anyURI </parent> ?
    <properties>
        <property key="xs:string"> xs:string </property> *
```
3.6.11 ProtocolSegmentCollection serialization

The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML.

This Resource shall be serialized as follows:

**JSON serialization:**

```json

{ "resourceURI": "http://schemas.dmtf.org/cimi/2/ProtocolSegmentCollection",
 "id": string,
 "updated": string,
 "parent": string,
 "count": number,
 "segments": [
  { "resourceURI": "http://schemas.dmtf.org/cimi/2/ProtocolSegment",
   "id": string,
   ... remaining ProtocolSegment attributes ...
  }, +
 ],
 "operations": [ { "rel": "add", "href": string } ? ]
 ... }

```

```xml

</properties> ?
<state> xs:string </state>
@protocol> xs:string </protocol>
<noDefaultRouting> xs:boolean </noDefaultRouting >
<endpoints href="xs:anyURI"/>
<parameters>
  <parameter key="xs:string"> xs:string </parameters> *
</parameters>
<meters href="xs:anyURI"/> ?
<eventLog" href="xs:anyURI"/> ?
<operations>
  <operation rel="edit" href="xs:anyURI"/ > ?
  <operation rel="delete" href="xs:anyURI"/ > ?
  <operation rel="http://schemas.dmtf.org/cimi/2/action/start" href="xs:anyURI"/ > ?
  <operation rel="http://schemas.dmtf.org/cimi/2/action/stop" href="xs:anyURI"/ > ?
</operations>
</ProtocolSegment>
```
XML serialization:

```xml
<Collection
  resourceURI="http://schemas.dmtf.org/cimi/2/ProtocolSegmentCollection"
  xmlns="http://schemas.dmtf.org/cimi/2">
  <id>xs:anyURI</id>
  <updated>xs:dateTime</updated>
  <parent>xs:anyURI</parent>
  <count>xs:integer</count>
  <segments>
    <ProtocolSegment>
      <id>xs:anyURI</id>
      ... remaining ProtocolSegment attributes ...
    </ProtocolSegment> *
  </segments>
  <operations>
    <operation rel="add" href="xs:anyURI"/> ?
  </operations>
  <xs:any>*
</Collection>
```

### 3.6.12 ProtocolSegmentTemplate serialization

The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML.

**JSON media type:** application/json

**JSON serialization:**

```json
{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/ProtocolSegmentTemplate",
  "id": string,
  "name": string, ?
  "description": string, ?
  "created": string, ?
  "updated": string, ?
  "parent": string, ?
  "properties": { string: string, + }, ?
  "network": { "href": string }, ?
  "initialState": string, ?
  "protocol": string,
  "noDefaultRouting": boolean,
  "endpoints": { "href": string }, *
  "endpointTemplates": [
      { "href": string, ?
```

```json
```
... ProtocolEndpointTemplate attributes ...
), *
}, ?,
.. parameters": [ string: string, + ], ?,
"meterTemplates": [

  /*
  "href": string, ?
  ... MeterTemplate attributes ... ?
  }, *
}, ?, 
"eventLogTemplate": {
  /*
  "href": string, ?
  ... EventLogTemplate attributes ... ?
  }, ?,
"operations": [
  { "rel": "edit", "href": string }, ?,
  { "rel": "delete", "href": string } ?
  ] ?,
  ...
} ?

**XML media type:** application/xml

**XML serialization:**

```xml
<ProtocolSegmentTemplate xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <name> xs:string </name> ?
  <description> xs:string </description> ?
  <created> xs:dateTime </created> ?
  <updated> xs:dateTime </updated> ?
  <parent> xs:anyURI </parent> ?
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties> ?
  <network href="xs:anyURI"/> ?
  <initialState> xs:string </initialState> ?
  <protocol> xs:string </protocol>
  <noDefaultRouting> xs:boolean </noDefaultRouting >
  <endpoints> *
    <ProtocolEndpoint href="xs:anyURI"/> *
  </endpoints>
  <endpointTemplates>
```
ProtocolEndpointTemplate attributes ...

<protocolTemplates> *
<parameters>
  <parameter key="xs:string"> xs:string </parameters> *
</parameters>
<meterTemplates>
  <MeterTemplate href="xs:anyURI"? >
    ... MeterTemplate attributes ...
  </MeterTemplate> *
</meterTemplates>
<eventLogTemplate href="xs:anyURI"? >
  ... EventLogTemplate attributes ...
</eventLogTemplate> *
<operations>
  <operation rel="edit" href="xs:anyURI"/> ?
  <operation rel="delete" href="xs:anyURI"/> ?
</operations> *
</ ProtocolSegmentTemplate>

3.6.13 ProtocolSegmentTemplateCollection serialization

The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML.

This Resource shall be serialized as follows:

**JSON serialization:**

```json
{ "resourceURI":
  "http://schemas.dmtf.org/cimi/2/ProtocolSegmentTemplateCollection",
  "id": string,
  "updated": string,
  "parent": string,
  "count": number,
  "protocolSegmentTemplates": [
    { "resourceURI": "http://schemas.dmtf.org/cimi/2/ProtocolSegmentTemplate",
      "id": string,
      ... remaining ProtocolSegmentTemplate attributes ...
    }, +
  ],
  "operations": [ { "rel": "add", "href": string } ? ]
```
XML serialization:

```xml
<Collection
    resourceURI="http://schemas.dmtf.org/cimi/2/ProtocolSegmentTemplateCollection"
    xmlns="http://schemas.dmtf.org/cimi/2">
    <id> xs:anyURI </id>
    <updated> xs:dateTime </updated>
    <parent> xs:anyURI </parent>
    <count> xs:integer </count>
    <protocolSegmentTemplates>
        <ProtocolSegmentTemplate>
            <id> xs:anyURI </id>
            ... remaining ProtocolSegmentTemplate attributes ...
        </ProtocolSegmentTemplate> *
    </protocolSegmentTemplates>
    <operations>
        <operation rel="add" href="xs:anyURI"/> ?
    </operations>
    <xs:any>*
</Collection>
```

3.6.14 ProtocolEndpoint serialization

The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML.

**JSON media type: application/json**

**JSON serialization:**

```json
{
    "resourceURI": "http://schemas.dmtf.org/cimi/2/ProtocolEndpoint",
    "id": string, 
    "name": string, ?
    "description": string, ?
    "created": string, ?
    "updated": string, ?
    "parent": string, ?
    "properties": { string: string, + }, ?
    "state": string,
    "protocol": string, ?
    "address": string,
    "origin": string,
    ...
}
```
"interface": { "href": string },
.."parameters": { string: string, + }, ?
"meters": { "href": string }, ?
"eventLog": { "href": string }, ?
"operations": [
  { "rel": "edit", "href": string }, ?
  { "rel": "delete", "href": string }, ?
  { "rel": "http://schemas.dmtf.org/cimi/2/action/enable", "href": string }, ?
  { "rel": "http://schemas.dmtf.org/cimi/2/action/disable", "href": string } ?
  ] ?
] ...

XML media type: application/xml

XML serialization:

```xml
<ProtocolEndpoint xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <name> xs:string </name> ?
  <description> xs:string </description> ?
  <created> xs:dateTime </created> ?
  <updated> xs:dateTime </updated> ?
  <parent> xs:anyURI </parent> ?
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties> ?
  <state> xs:string </state>
  <protocol> xs:string </protocol> ?
  <address> xs:string </address>
  <origin> xs:string </origin>
  <interface href="xs:anyURI"/>
  <parameters>
    <parameters key="xs:string"> xs:string </parameters> *
  </parameters>
  <meters href="xs:anyURI"/> ?
  <eventLog" href="xs:anyURI"/> ?
  <operations>
    <operation rel="edit" href="xs:anyURI"/> ?
    <operation rel="delete" href="xs:anyURI"/> ?
    <operation rel="http://schemas.dmtf.org/cimi/2/action/enable" href="xs:anyURI"/> ?
```
3.6.15 ProtocolEndpointCollection serialization

The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML.

This Resource shall be serialized as follows:

**JSON serialization:**

```json
{
    "resourceURI": "http://schemas.dmtf.org/cimi/2/ProtocolEndpointCollection",
    "id": string,
    "updated": string,
    "parent": string,
    "count": number,
    "endpoints": [
        {
            "resourceURI": "http://schemas.dmtf.org/cimi/2/ProtocolEndpoint",
            "id": string,
            ... remaining ProtocolEndpoint attributes ...
        }, *
    ], ?
    "operations": [ { "rel": "add", "href": string } ? ]
}
```

**XML serialization:**

```xml
<Collection
    resourceURI="http://schemas.dmtf.org/cimi/2/ProtocolEndpointCollection"
    xmlns="http://schemas.dmtf.org/cimi/2">
    <id> xs:anyURI </id>
    <updated> xs:dateTime </updated>
    <parent> xs:anyURI </parent>
    <count> xs:integer </count>
    <endpoints>
        <ProtocolEndpoint>
            <id> xs:anyURI </id>
            ... remaining ProtocolEndpoint attributes ...
        </ProtocolEndpoint> *
    </endpoints>
    <operations>
```

<operation rel="http://schemas.dmtf.org/cimi/2/action/disable"
href="xs:anyURI"/> ?

</ProtocolEndpoint>
3.6.16 ProtocolEndpointTemplate serialization

The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML.

Both Consumer and Provider shall serialize this Resource as described below. The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML.

**JSON media type**: application/json

**JSON serialization**:

```json
{
    "resourceURI": "http://schemas.dmtf.org/cimi/2/ProtocolEndpointTemplate",
    "id": string,
    "name": string, ?
    "description": string, ?
    "created": string, ?
    "updated": string, ?
    "parent": string, ?
    "properties": { string: string, + }, ?
    "initialState": string, ?
    "address": string, ?
    "origin": string,
    "interface": { "href": string }, ?
    "parameters": { string: string, + }, ?
    "meterTemplates": [ 
        { "href": string, ?
            ... MeterTemplate attributes ... ?
        }, *
    ], ?,
    "eventLogTemplate": { 
        "href": string, ?
        ... EventLogTemplate attributes ... ?
    }, ?,
    "operations": [ 
        { "rel": "edit", "href": string }, ?
        { "rel": "delete", "href": string } ?
    ] ?
}
```
XML media type: application/xml

XML serialization:

```xml
<ProtocolEndpointTemplate xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <name> xs:string </name> ?
  <description> xs:string </description> ?
  <created> xs:dateTime </created> ?
  <updated> xs:dateTime </updated> ?
  <parent> xs:anyURI </parent> ?
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties> ?
  <initialState> xs:string </initialState> ?
  <address> xs:string </address> ?
  <origin> xs:string </origin>
  <interface href="xs:anyURI"/> ?
  <parameters key="xs:string"> xs:string </parameters> *
  <meterTemplates>
    <MeterTemplate href="xs:anyURI"/> *
    ... MeterTemplate attributes ... ?
  </MeterTemplate>
  <eventLogTemplate href="xs:anyURI"/> ?
  ... EventLogTemplate attributes ... ?
</eventLogTemplate> ?
<operations>
  <operation rel="edit" href="xs:anyURI"/> ?
  <operation rel="delete" href="xs:anyURI"/> ?
</operations>
<xs:any>*
</ProtocolEndpointTemplate>
```

### 3.6.17 ProtocolEndpointTemplateCollection serialization

The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML.

This Resource shall be serialized as follows:

**JSON serialization:**

```json
{ "resourceURI":
  "http://schemas.dmtf.org/cimi/2/ProtocolEndpointTemplateCollection",
  "id": string,
```
3.6.18 NetworkInterface serialization

The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML.

**JSON media type:** application/json

**JSON serialization:**

```json
  { "resourceURI": "http://schemas.dmtf.org/cimi/2/NetworkInterface",
    "id": string,
    ...
  }
```

**XML serialization:**

```xml
  <Collection
    resourceURI="http://schemas.dmtf.org/cimi/2/NetworkInterfaceCollection"
    xmlns="http://schemas.dmtf.org/cimi/2">
    <id> xs:anyURI </id>
    <updated> xs:dateTime </updated>
    <parent> xs:anyURI </parent>
    <count> xs:integer </count>
    <protocolEndpointTemplates>
      <ProtocolEndpointTemplate>
        <id> xs:anyURI </id>
        ...
      </ProtocolEndpointTemplate> *
    </protocolEndpointTemplates>
    <operations>
      <operation rel="add" href="xs:anyURI"/> ?
    </operations>
    <xs:any>*
  </Collection>
```
"name": string, ?
"description": string, ?
"created": string, ?
"updated": string, ?
"parent": string, ?
"properties": { string: string, + }, ?
"state": string,
"endpoints": { "href": string }, ?
"speed": number, ?
"mtu": number ?,
"meters": { "href": string }, ?
"eventLog": { "href": string }, ?
"operations": [
  { "rel": "edit", "href": string }, ?
  { "rel": "delete", "href": string }, ?
  { "rel": "http://schemas.dmtf.org/cimi/2/action/enable", "href": string },
  ?
  { "rel": "http://schemas.dmtf.org/cimi/2/action/disable", "href": string }
  ?
],
...}
3.6.19 NetworkInterfaceCollection serialization

The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML.

This Resource shall be serialized as follows:

**JSON serialization:**

```json
{  "resourceURI": "http://schemas.dmtf.org/cimi/2/NetworkInterfaceCollection",  "id": string,  "updated": string,  "parent": string,  "count": number,  "interfaces": [    {  "resourceURI": "http://schemas.dmtf.org/cimi/2/NetworkInterface",  "id": string,    ... remaining NetworkInterface attributes ...    }, +  },  "operations": [ {  "rel": "add",  "href": string } ? ]  ... }
```

**XML serialization:**

```xml
<Collection resourceURI="http://schemas.dmtf.org/cimi/2/NetworkInterfaceCollection"
            xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <updated> xs:dateTime </updated>
  <parent> xs:anyURI </parent>
  <count> xs:integer </count>
  <interfaces>
    <NetworkInterface>
      <id> xs:anyURI </id>
    </NetworkInterface>
  </interfaces>
</Collection>
```
3.6.20 NetworkInterfaceTemplate serialization

The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML.

Both Consumer and Provider shall serialize this Resource as described below. The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML.

**JSON media type:** application/json

**JSON serialization:**

```json
{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/NetworkInterfaceTemplate",
  "id": string,
  "name": string, ?
  "description": string, ?
  "created": string, ?
  "updated": string, ?
  "parent": string, ?
  "properties": { string: string, + }, ?
  "initialState": string, ?
  "endpoints": { "href": string }, ?,
  "speed": number, ?
  "mtu": number ?,
  "meterTemplates": [
    { "href": string, ?
      "... MeterTemplate attributes ... ?
    }, *
  ], ?
  "eventLogTemplate": {
    "href": string, ?
    "... EventLogTemplate attributes ... ?
  }, ?
  "operations": [
    { "rel": "edit", "href": string }, ?
    { "rel": "delete", "href": string } ?
  ]
}
```
The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML.

This Resource shall be serialized as follows:
JSON serialization:

```json
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/NetworkInterfaceTemplateCollection",
  "id": string,
  "updated": string,
  "parent": string,
  "count": number,
  "networkInterfaceTemplates": [
    { "resourceURI": "http://schemas.dmtf.org/cimi/2/NetworkInterfaceTemplate",
      "id": string,
      ... remaining NetworkInterfaceTemplate attributes ...
    }, +
  ],
  "operations": [ { "rel": "add", "href": string } ? ]
}
```

XML serialization:

```xml
<Collection
  resourceURI="http://schemas.dmtf.org/cimi/2/NetworkInterfaceTemplateCollection"
  xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <updated> xs:dateTime </updated>
  <parent> xs:anyURI </parent>
  <count> xs:integer </count>
  <networkInterfaceTemplates>
    <NetworkInterfaceTemplate>
      <id> xs:anyURI </id>
      ... remaining NetworkInterfaceTemplate attributes ...
    </NetworkInterfaceTemplate> *
  </networkInterfaceTemplates>
  <operations>
    <operation rel="add" href="xs:anyURI"/>
  </operations>
</Collection>
```

3.7 Monitoring Resources group

3.7.1 Monitoring Resources and their relationships

The group of Monitoring-related Resources is represented in Figure 13, with other associated Resources:
3.7.2 Job serialization

The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML.

**JSON media type**: application/json

**JSON serialization**:

```json
{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/Job",
  "id": string,
  "name": string, ?
  "description": string, ?
  "created": string, ?
  "updated": string, ?
  "parent": string, ?
  "properties": { string: string, + }, ?
  "state": string,
  "targetResource": { "href": string },
  "affectedResources": [ { "href": string }, + ],
  "action": string,
  "returnCode": number,
...}
```
Cloud Infrastructure Management Interface (CIMI) Primer

XML media type: application/xml

XML serialization:

```xml
<Job xmlns="http://schemas.dmtf.org/cimi/2">
  <id xs:anyURI /></id>
  <name xs:string /></name> ?
  <description xs:string /></description> ?
  <created xs:dateTime /></created> ?
  <updated xs:dateTime /></updated> ?
  <parent xs:anyURI /></parent> ?
  <properties>
    <property key="xs:string"> xs:string /></property> *
  </properties> ?
  <state xs:string /></state>
  <targetResource href="xs:anyURI"/>
  <affectedResources>
    <affectedResource href="xs:anyURI"/> +
  </affectedResources>
  <action xs:anyURI /></action>
  <returnCode xs:integer /></returnCode>
  <progress xs:integer /></progress>
  <statusMessage xs:string /></statusMessage>
  <timeOfStatusChange xs:dateTime /></timeOfStatusChange>
  <parentJob href="xs:anyURI"/> ?
  <nestedJobs>
    <nestedJob href="xs:anyURI"/> *
</Job>
```
3.7.3 **JobCollection serialization**

The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML.

This Resource shall be serialized as follows:

**JSON serialization:**

```json
{
   "resourceURI": "http://schemas.dmtf.org/cimi/2/JobCollection",
   "id": string,
   "updated": string,
   "parent": string,
   "count": integer,
   "jobs": [
      {
         "resourceURI": "http://schemas.dmtf.org/cimi/2/Job",
         "id": string,
         ... remaining Job attributes ...
      }, +
   ] ?
...}
```

**XML serialization:**

```xml
<Collection resourceURI="http://schemas.dmtf.org/cimi/2/JobCollection"
   xmlns="http://schemas.dmtf.org/cimi/2">
   <id> xs:anyURI </id>
   <updated> xs:dateTime </updated>
   <parent> xs:anyURI </parent>
   <count> xs:integer </count>
   <jobs>
      <Job>
         <id> xs:anyURI </id>
         ... remaining Job attributes ...
      </Job> *
   </jobs>
</Collection>
```
3.7.4 Meter serialization

The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML.

**JSON media type**: application/json

**JSON serialization**:

```json
{  
  "resourceURI": "http://schemas.dmtf.org/cimi/2/Meter",
  "id": string,
  "name": string, ?
  "description": string, ?
  "created": string, ?
  "updated": string, ?
  "parent": string, ?
  "properties": {  string: string, + }, ?,
  "targetResource": {  "href": string },
  "aspect": string,
  "units": string,
  "sampleInterval": number,
  "timeScope": string,
  "intervalDuration": string,
  "isContinuous": boolean,
  "samples": {  "href": string }, ?,
  "minValue": string, ?
  "maxValue": string, ?
  "stopTime": string, ?
  "expiresTime": string, ?
  "operations": [
    {  "rel": "edit", "href": string }, ?,
    {  "rel": "delete", "href": string }, ?,
    {  "rel": "http://schemas.dmtf.org/cimi/2/action/start", "href": string }, ?,
    {  "rel": "http://schemas.dmtf.org/cimi/2/action/stop", "href": string } ?
  ]
...
}
```
XML media type: application/xml

XML serialization:

```xml
<Meter xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <name> xs:string </name> ?
  <description> xs:string </description> ?
  <created> xs:dateTime </created> ?
  <updated> xs:dateTime </updated> ?
  <parent> xs:anyURI </parent> ?
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties> ?
  <targetResource href="xs:anyURI"/>
  <aspect> xs:anyURI </aspect>
  <units> xs:string </units>
  <sampleInterval> xs:integer </sampleInterval>
  <timeScope> xs:string <timeScope>
  <intervalDuration xs:duration </intervalDuration>
  <isContinuous> xs:boolean </isContinuous>
  <samples href="xs:anyURI"/> ?
  <minValue> xs:string </minValue> ?
  <maxValue> xs:string </maxValue> ?
  <stopTime> xs:dateTime </stopTime> ?
  <expiresTime> xs:dateTime </expiresTime> ?
  <operations>
    <operation rel="edit" href="xs:anyURI"/> ?
    <operation rel="delete" href="xs:anyURI"/> ?
    <operation rel="http://schemas.dmtf.org/cimi/2/action/start" href="xs:anyURI"/> ?
    <operation rel="http://schemas.dmtf.org/cimi/2/action/stop" href="xs:anyURI"/> ?
  </operations>
  <xs:any>*
</Meter>
```

3.7.5 MeterCollection serialization

The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML.

This Resource shall be serialized as follows:
3.7.6 SampleCollection serialization

The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML.

**JSON serialization:**

```json
{
   "resourceURI": "http://schemas.dmtf.org/cimi/2/MeterCollection",
   "id": string,
   "updated": string,
   "parent": string,
   "count": number,
   "meters": [
      {
         "resourceURI": "http://schemas.dmtf.org/cimi/2/Meter",
         "id": string,
         "... remaining Meter attributes ...
      }, +
   ],
   "operations": [ { "rel": "add", "href": string } ]
}
```

**XML serialization:**

```xml
<Collection resourceURI="http://schemas.dmtf.org/cimi/2/MeterCollection"
   xmlns="http://schemas.dmtf.org/cimi/2">
   <id> xs:anyURI </id>
   <updated> xs:dateTime </updated>
   <parent> xs:anyURI </parent>
   <count> xs:integer </count>
   <meters>
      <Meter>
         <id> xs:anyURI </id>
         ... remaining Meter attributes ...
      </Meter>*
   </meters>
   <operations>
      <operation rel="add" href="xs:anyURI"/> ?
   </operations>
   <xs:any>*
</Collection>
```

The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML.
"updated": string,
"parent": string,
"count": number,
"samples": [
    { "resourceURI": "http://schemas.dmtf.org/cimi/2/Sample",
      "id": string,
      "name": string, 
      "description": string, 
      "created": string, 
      "updated": string, 
      "properties": { string: string, + }, 
      "timestamp": string,
      "value": string
      ... 
    }, +
    ], 
...

XML serialization:

```xml
<Collection
    resourceURI="http://schemas.dmtf.org/cimi/2/SampleCollection"
    xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <count> xs:integer </count>
  <samples>
    <Sample>
      <id> xs:anyURI </id>
      <name> xs:string </name> 
      <description> xs:string </description> 
      <created> xs:dateTime </created> 
      <updated> xs:dateTime </updated> 
      <properties>
        <property key="xs:string"> xs:string </property> *
      </properties> 
      <sample timestamp="xs:dateTime" value="xs:string"/>
    </Sample>*
  </samples>
</Collection>
```
3.7.7 MeterTemplate serialization

The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML.

**JSON media type**: application/json

**JSON serialization**:
```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/MeterTemplate",
  "id": string, 
  "name": string, 
  "description": string, 
  "created": string, 
  "updated": string, 
  "parent": string, 
  "properties": { string: string, + }, 
  "targetResource": { string }, 
  "meterConfig": {
    "href": string | ... MeterConfiguration attributes ...
  },
  "operations": [
    { "rel": "edit", "href": string }, 
    { "rel": "delete", "href": string }
  ]
}
```

**XML media type**: application/xml

**XML serialization**:
```
<MeterTemplate xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <name> xs:string </name> 
  <description> xs:string </description> 
  <created> xs:dateTime </created> 
  <updated> xs:dateTime </updated> 
  <parent> xs:anyURI </parent> 
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties> 
  <targetResource href="xs:anyURI"/>
  <meterConfig href="xs:anyURI">
    ... MeterConfiguration attributes ...
  </meterConfig>
</MeterTemplate>
```
3.7.8 MeterTemplateCollection serialization

The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML.

This Resource shall be serialized as follows:

**JSON serialization:**

```json
{
    "resourceURI": "http://schemas.dmtf.org/cimi/2/MeterTemplateCollection",
    "id": string,
    "updated": string,
    "parent": string,
    "count": number,
    "meterTemplates": [
        {
            "resourceURI": "http://schemas.dmtf.org/cimi/2/MeterTemplate",
            "id": string,
            ... remaining MeterTemplate attributes ...
        }, +
    ],
    "operations": [ { "rel": "add", "href": string } ]
}
```

**XML serialization:**

```xml
<Collection
    resourceURI="http://schemas.dmtf.org/cimi/2/MeterTemplateCollection"
    xmlns="http://schemas.dmtf.org/cimi/2">
    <id> xs:anyURI </id>
    <updated> xs:dateTime </updated>
    <parent> xs:anyURI </parent>
    <count> xs:integer </count>
    <meterTemplates>
        <MeterTemplate>
            <id> xs:anyURI </id>
            ... remaining MeterTemplate attributes ...
        </MeterTemplate> *
    </meterTemplates>
</Collection>
```
Cloud Infrastructure Management Interface (CIMI) Primer

<operations>
    <operation rel="add" href="xs:anyURI"/>
</operations>

</Collection>

3.7.9 MeterConfiguration serialization

The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML.

**JSON media type:** application/json

**JSON serialization:**

```json
{
    "resourceURI": "http://schemas.dmtf.org/cimi/2/MeterConfiguration",
    "id": string,
    "name": string, ?
    "description": string, ?
    "created": string, ?
    "updated": string, ?
    "parent": string, ?
    "properties": { string: string, + }, ?
    "associatedResources": [
        { "href": string }, +
    ], ?
    "aspect": string,
    "units": string,
    "sampleInterval": number,
    "timeScope": string,
    "intervalDuration": string,
    "isContinuous": boolean,
    "operations": [
        { "rel": "edit", "href": string }, ?
        { "rel": "delete", "href": string } ?
    ]
}
```

**XML media type:** application/xml

**XML serialization:**

```xml
<MeterConfiguration xmlns="http://schemas.dmtf.org/cimi/2">
    <id> xs:anyURI </id>
    <name> xs:string </name>
</MeterConfiguration>
```
3.7.10 MeterConfigurationCollection serialization

The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML.

This Resource shall be serialized as follows:

**JSON serialization:**

```json
{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/MeterConfigurationCollection",
  "id": string,
  "updated": string,
  "parent": string,
  "count": number,
  "meterConfigurations": [
    {
      "resourceURI": "http://schemas.dmtf.org/cimi/2/MeterConfiguration",
      "id": string,
      ... remaining MeterConfiguration attributes ...
    }, +
  ],
  "operations": [ { "rel": "add", "href": string } ? ]
}
```
XML serialization:

```
<Collection
    resourceURI="http://schemas.dmtf.org/cimi/2/MeterConfigurationCollection"
    xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <updated> xs:dateTime </updated>
  <parent> xs:anyURI </parent>
  <count> xs:integer </count>
  <meterConfigurations>
    <MeterConfiguration>
      <id> xs:anyURI </id>
      ... remaining MeterConfiguration attributes ...
    </MeterConfiguration> *
  </meterConfigurations>
  <operations>
    <operation rel="add" href="xs:anyURI"/> ?
  </operations>
  <xs:any>*
</Collection>
```

### 3.7.11 Event serialization

The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML.

**JSON media type**: application/json

**JSON serialization**:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/Event",
  "id": string,
  "name": string, ?
  "description": string, ?
  "created": string, ?
  "updated": string, ?
  "parent": string, ?
  "properties": { string: string, + }, ?
  "timestamp": string,
  "type": string,
  "content": any, ?
  "outcome": string, ?
  "severity": string, ?
```
"contact": string, ?
...
}

XML media type: application/xml

XML serialization:

```xml
<Event xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <name> xs:string </name> ?
  <description> xs:string </description> ?
  <created> xs:dateTime </created> ?
  <updated> xs:dateTime </updated> ?
  <parent> xs:anyURI </parent> ?
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties> ?
  <timestamp> xs:dateTime </timestamp>
  <type> xs:string </type>
  <content> xs:any* </content> ?
  <outcome> xs:string </outcome> ?
  <severity> xs:string </severity> ?
  <contact> xs:string </contact> ?
  <xs:any>*
</Event>
```

The following pseudo-schemas describe the serialization of the "content" property for various types of events:

"state" event:

JSON serialization:

```json
{
  "id": string,
  ...
  "type": "http://schemas.dmtf.org/cimi/2/event/state",
  "content": {
    "resName": string,
    "resource": { "href": string },
    "resType": string,
    "state": string,
    "previous": string ?
  }
}
```
XML serialization:

```
<Event xmlns="http://schemas.dmtf.org/cimi/2">
  ...<type> http://schemas.dmtf.org/cimi/2/event/state </type>
  <content>
    <resName> xs:string </resName>
    <resource href="xs:anyURI"/>
    <resType> xs:anyURI </resType>
    <state> xs:string </state>
    <previous> xs:string </previous> ?
  </content> ?
  ...
</Event>
```

"alarm" event:

JSON serialization:

```
{
  "id": string,
  ...
  "type": "http://schemas.dmtf.org/cimi/2/event/alarm",
  "content": {
    "resName": string ?,
    "resource": { "hhref": string }, ?,
    "resType": string ?,
    "code": string,
    "detail": string ?
  }
}
```

XML serialization:

```
<Event xmlns="http://schemas.dmtf.org/cimi/2">
  ...<type> http://schemas.dmtf.org/cimi/2/event/alarm </type>
  <content>
    <resname> xs:string </resname> ?
    <resource href="xs:anyURI"/> ?
    <restype> xs:anyURI </restype> ?
    <code> xs:string </code>
    <detail> xs:string </detail> ?
  </content> ?
</Event>
```
"model" event:

**JSON serialization:**

```json
{
  "id": string,
  ...
  "type": "http://schemas.dmtf.org/cimi/2/event/model",
  "content": {
    "resName": string, ?
    "resource": { "href": string }, ?
    "resType": string, ?
    "change": string,
    "detail": string ?
  }
}
```

**XML serialization:**

```xml
<Event xmlns="http://schemas.dmtf.org/cimi/2">
  ...
  <type> http://schemas.dmtf.org/cimi/2/event/model </type>
  <content>
    <resname> xs:string </resname> ?
    <resource href="xs:anyURI"/> ?
    <restype> xs:anyURI </restype> ?
    <change> xs:string </change>
    <detail> xs:string </detail> ?
  </content> ?
  ...
</Event>
```
"access" event:

JSON serialization:

```json
{
  "id": string,
  ...
  "type": "http://schemas.dmtf.org/cimi/2/event/access",
  "content": {
    "operation": string,
    "resource": { "href": string },
    "detail": string, ?
    "initiator": string ?
  }
  ...
}
```

XML serialization:

```xml
<Event xmlns="http://schemas.dmtf.org/cimi/2">
  ...
  <type> http://schemas.dmtf.org/cimi/2/event/access </type>
  <content>
    <operation> xs:string </operation>
    <resource href="xs:anyURI"/>
    <detail> xs:string </detail> ?
    <initiator> xs:string </initiator> ?
  </content> ?
  ...
</Event>
```

3.7.12 EventLog serialization

The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML.

**JSON media type:** application/json

**JSON serialization:**

```json
{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/EventLog",
  "id": string,
  "name": string, ?
  "description": string, ?
  "created": string, ?
  "updated": string, ?
  "parent": string, ?
  "properties": { string: string, + }, ?
}
XML serialization:

```xml
<EventLog xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <name> xs:string </name> ?
  <description> xs:string </description> ?
  <created> xs:dateTime </created> ?
  <updated> xs:dateTime </updated> ?
  <parent> xs:anyURI </parent> ?
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties> ?
  <targetResource href="xs:anyURI"/>
  <events href="xs:anyURI"/>
  <persistence> xs:string </persistence>
  <summary>
    <low> xs:integer </low>
    <medium> xs:integer </medium>
    <high> xs:integer </high>
    <critical> xs:integer </critical>
  </summary>
  <operations>
    <operation rel="edit" href="xs:anyURI"/> ?
    <operation rel="delete" href="xs:anyURI"/> ?
</EventLog>
```
3.7.13 EventCollection serialization

The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML.

**JSON serialization:**

```json
{
    "resourceURI": "http://schemas.dmtf.org/cimi/2/EventCollection",
    "id": string,
    "updated": string,
    "parent": string,
    "count": number,
    "events": [
      {
        "resourceURI": "http://schemas.dmtf.org/cimi/2/Event",
        "id": string,
        ... remaining Event attributes ...
      }, +
    ],
    "operations": [ { "rel": "add", "href": string } ? ]
}
```

**XML serialization:**

```xml
<Collection resourceURI="http://schemas.dmtf.org/cimi/2/EventCollection"
             xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <updated> xs:dateTime </updated>
  <parent> xs:anyURI </parent>
  <count> xs:integer </count>
  <events>
    <Event>
      <id> xs:anyURI </id>
      ... remaining Event attributes ...
    </Event> *
  </events>
  <operations>
    <operation rel="add" href="xs:anyURI"/> ?
  </operations>
  <xs:any>*
</Collection>
```
3.7.14 EventLogCollection serialization

The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML.

This Resource shall be serialized as follows:

**JSON serialization:**

```json
{
    "resourceURI": "http://schemas.dmtf.org/cimi/2/EventLogCollection",
    "id": string,
    "updated": string,
    "parent": string,
    "count": number,
    "eventLogs": [
        { "resourceURI": "http://schemas.dmtf.org/cimi/2/EventLog",
        "id": string,
        ... remaining EventLog attributes ...
    ],
    "operations": [ { "rel": "add", "href": string } ]
}
```

**XML serialization:**

```xml
<EventLogCollection resourceURI="http://schemas.dmtf.org/cimi/2/EventLogCollection"
   xmlns="http://schemas.dmtf.org/cimi/2">
    <id> xs:anyURI </id>
    <updated> xs:dateTime </updated>
    <parent> xs:anyURI </parent>
    <count> xs:integer </count>
    <eventLogs>
        <EventLog>
            <id> xs:anyURI </id>
            ... remaining EventLog attributes ...
        </EventLog> *
    </eventLogs>
    <operations>
        <operation rel="add" href="xs:anyURI"/>
    </operations>
    <xs:any>*
</Collection>
```
3.7.15 EventLogTemplate serialization

The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML.

**JSON media type:** application/json

**JSON serialization:**

```json
{
    "resourceURI": "http://schemas.dmtf.org/cimi/2/EventLogTemplate",
    "id": string,
    "name": string, ?
    "description": string, ?
    "created": string, ?
    "updated": string, ?
    "parent": string, ?
    "properties": { string: string, + }, ?
    "targetResource": { string },
    "persistence": string,
    "operations": [
        { "rel": "edit", "href": string }, ?
        { "rel": "delete", "href": string } ?
    ] ?
}
```

**XML media type:** application/xml

**XML serialization:**

```xml
<EventLogTemplate xmlns="http://schemas.dmtf.org/cimi/2">
    <id> xs:anyURI </id>
    <name> xs:string </name> ?
    <description> xs:string </description> ?
    <created> xs:dateTime </created> ?
    <updated> xs:dateTime </updated> ?
    <parent> xs:anyURI </parent> ?
    <properties>
        <property key="xs:string"> xs:string </property> *
    </properties> ?
    <targetResource href="xs:anyURI"/>
    <persistence> xs:string </persistence>
    <operations>
        <operation rel="edit" href="xs:anyURI"/> ?
        <operation rel="delete" href="xs:anyURI"/> ?
    </operations>
</EventLogTemplate>
```
3.7.16 EventLogTemplateCollection serialization

The following pseudo-schemas (see notation in 1.3) describe the serialization of the Resource in both JSON and XML.

This Resource shall be serialized as follows:

**JSON serialization:**

```json
{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/EventLogTemplateCollection",
  "id": string,
  "updated": string,
  "parent": string,
  "count": number,
  "eventLogTemplates": [
    {
      "resourceURI": "http://schemas.dmtf.org/cimi/2/EventLogTemplate",
      "id": string,
      ... remaining EventLogTemplate attributes ...
    }, +
  ],
  "operations": [ { "rel": "add", "href": string } ]
}
```

**XML serialization:**

```xml
<Collection
  resourceURI="http://schemas.dmtf.org/cimi/2/EventLogTemplateCollection"
  xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <updated> xs:dateTime </updated>
  <parent> xs:anyURI </parent>
  <count> xs:integer </count>
  <eventLogTemplates>
    <EventLogTemplate>
      <id> xs:anyURI </id>
      ... remaining EventLogTemplate attributes ...
    </EventLogTemplate> *
  </eventLogTemplates>
  <operations>
    <operation rel="add" href="xs:anyURI"/>
  </operations>
</Collection>
```
3.8 Other Examples of serialized collections

The following are examples of how collections of Resources are serialized. The first one illustrates a basic collection of Machine resources. The second one illustrates an “enhanced” collection of Volume resources, i.e., with additional (accessory) attributes for each Volume item, besides those defined in the Volume resource type.

3.8.1 Machine Collection

The Resource type for each item of this Collection is ”Machine”. There is no accessory attributes in this Collection, which is then called a “basic” Machine Collection. In the example below, each Machine item in the Collection is not expanded except for its common attributes. An expanded serialization showing all or parts of each Machine is also an option – this may vary depending on what selection has been used when querying Resources of this type (subsetting with $select attribute).

JSON serialization:

```json
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/MachineCollection",
  "id": string,
  "updated": string,
  "parent": string,
  "count": number,
  "Machines": [
    { "resourceURI": "http://schemas.dmtf.org/cimi/2/Machine",
      "id": string,
      "name": string,
      "description": string,
      "created": string,
      "updated": string,
      "parent": string,
      "properties": { string: string, + },
      "machine": { "href": string },
      "operations": [
        { "rel": "edit", "href": string },
        { "rel": "delete", "href": string }
      ]
    }, ...
  ],
  "operations": [
    { "rel": "add", "href": string }
    { "rel": "insert", "href": string }
    { "rel": "remove", "href": string }
  ]
}
```
XML serialization:

```
<Collection
    resourceURI="http://schemas.dmtf.org/cimi/2/MachineCollection"
    xmlns="http://schemas.dmtf.org/cimi/2">
    <id> xs:anyURI </id>
    <updated> xs:dateTime </updated>
    <parent> xs:anyURI </parent>
    <count> xs:integer </count>
    <Machines>
        <Machine>
            <id> xs:anyURI </id>
            <name> xs:string </name>
            <description> xs:string </description>
            <created> xs:dateTime </created>
            <updated> xs:dateTime </updated>
            <parent> xs:anyURI </parent>
            <property key="xs:string"> xs:string </property> *
            <machine href="xs:anyURI"/>
            <operation rel="edit" href="xs:anyURI"/>
            <operation rel="delete" href="xs:anyURI"/>
        </Machine> *
    </Machines>
    <operation rel="add" href="xs:anyURI"/>
    <operation rel="insert" href="xs:anyURI"/>
    <operation rel="remove" href="xs:anyURI"/>
    <xs:any>*
</Collection>
```

3.8.2 Volume Collection in a Machine

The Resource type for each item of this Collection is “Volume”. The initial location of these Volumes (not part of Volume attributes) is added as an accessory attribute to each Collection item. In the example below, each Volume item in the Collection is not expanded except for its common attributes. An expanded serialization showing all or parts of each Volume is also an option.

Note that the last part of the Collection resourceURI is not just `VolumeCollection` but `locatedVolumeCollection`, in order to denote the addition of an accessory attribute `initialLocation` to each Resource item of type `Volume`. 
JSON serialization:

```json
{
  "resourceURI": "http://schemas.dmtf.org/cimi/2/locatedVolumeCollection",
  "id": string,
  "updated": string,
  "parent": string,
  "count": number,
  "locatedVolumes": [
    {
      "resourceURI": "http://schemas.dmtf.org/cimi/2/locatedVolume",
      "id": string,
      "name": string,
      "description": string,
      "created": string,
      "updated": string,
      "parent": string,
      "properties": { string: string, + },
      "initialLocation": string,
      "volume": { "href": string },
      "operations": ["edit", "delete"]
    }, ...
  ],
  "operations": [
    "add", "insert", "remove"
  ]
}
```

XML serialization:

```xml
<Collection
  resourceURI="http://schemas.dmtf.org/cimi/2/locatedVolumeCollection"
  xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <updated> xs:dateTime </updated>
  <parent> xs:anyURI </parent>
  <count> xs:integer </count>
```

Cloud Infrastructure Management Interface (CIMI) Primer

DSP2027

Published Version 2.0.0
<locatedVolumes>
  <locatedVolume>
    <id> xs:anyURI </id>
    <name> xs:string </name>
    <description> xs:string </description>
    <created> xs:dateTime </created>
    <updated> xs:dateTime </updated>
    <parent> xs:anyURI </parent>
    <property key="xs:string"> xs:string </property> *
    <initialLocation> xs:string </initialLocation>
    <volume href="xs:anyURI"/>
    <operations>
      <operation rel="edit" href="xs:anyURI"/>
      <operation rel="delete" href="xs:anyURI"/>
    </operations>
    <xs:any>*
  </locatedVolume> *
</locatedVolumes>
<operations>
  <operation rel="add" href="xs:anyURI"/>
  <operation rel="insert" href="xs:anyURI"/>
  <operation rel="remove" href="xs:anyURI"/>
</operations>
<xs:any>*
</Collection>
# Change log

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0.0</td>
<td>2015-08-28</td>
<td></td>
</tr>
<tr>
<td>1.0.1</td>
<td>2015-09-12</td>
<td>Errata</td>
</tr>
<tr>
<td>1.1.0</td>
<td>2014-06-09</td>
<td>This version contains the following updates:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• include network scenario</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• updated to old version of the network writeup</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• updated to include clause 8 for Mantis 2347 and Mantis 2095</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Dies editorial comments, fixed future tense.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Arturo updates folded in.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• delete subclause on Add Network Port; fixed example in 7.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Output of first editing session on June 2, 2014</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Dies added a network example</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Output of last editing session on June 9, 2014</td>
</tr>
<tr>
<td>2.0.0</td>
<td>2016-08-04</td>
<td>Upgraded to match CIMI 2.0 changes, added model information (section 2), imported pseudo-schema for Resource serialization from the CIMI specification (section 3).</td>
</tr>
</tbody>
</table>