Document Identifier: DSP0263

Date: 2016-03-22

Version: 2.0.0d

Cloud Infrastructure Management Interface (CIMI) Model and RESTful HTTP-based Protocol

An Interface for Managing Cloud Infrastructure

Supersedes: 1.0.1

Document Class: Normative

Document Status: Work in Progress

Document Language: en-US

CONTENTS

Tables

Foreword

The *Cloud Infrastructure Management Interface (CIMI) Model and RESTful HTTP-based Protocol* specification (DSP0263) was prepared by the DMTF Cloud Management Working Group. It defines a logical model for the management of resources within the Infrastructure as a Service domain.

DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability.

Acknowledgments

The DMTF acknowledges the following individuals for their contributions to this document:

Editors (present and past):

- Jacques Durand – Fujitsu
- Marios Andreou – Red Hat (previous)
- Doug Davis – IBM (previous)
- Gilbert Pilz – Oracle (previous)

Contributors:

- Ghazanfar Ali – ZTE Corporation
- Marios Andreou – Red Hat
- Keith Bankston – Microsoft Corporation
- Winston Bumpus – VMware Inc.
- Nathan Burkhart – Microsoft Corporation
- Mark Carlson – Oracle
- Steve Carter – Novell
- Junsheng Chu – ZTE Corporation
- Josh Cohen – Microsoft Corporation
- Derek Coleman – Hewlett-Packard Company
- John Crandall – Brocade Communications Systems
- Doug Davis – IBM
- Jim Davis – WBEM Solutions
- Fernando de la Iglesia – Telefónica
- Hiroshi Dempo – NEC Corporation
- Jacques Durand – Fujitsu
- Yigal Edery – Microsoft Corporation
- George Ericson – EMC
- Colleen Evans – Microsoft Corporation
- Norbert Floeren – Ericsson AB
- Robert Freund – Hitachi, Ltd.
- Fermín Galán – Telefónica
- Krishnan Gopalan – Microsoft Corporation
- Kazunori Iwasa – Fujitsu
- Mark Johnson – IBM
- Bhumip Khasnabish – ZTE Corporation
- Dies Köper – Fujitsu
- Vincent Kowalski – BMC Software
- Ruby Krishnaswamy – France Telecom Group
- Lawrence Lamers – VMware Inc.
- Paul Lipton – CA Technologies
- James Livingston – NEC Corporation
- Vince Lubsey – Virtustream Inc.

- David Lutterkort – Red Hat
- Fred Maciel – Hitachi, Ltd.
- Andreas Maier – IBM
- Ashok Malhotra – Oracle
- Arturo Martin de Nicolas - Ericsson
- Jeff Mischkinsky – Oracle
- Jesus Molina – Fujitsu
- Efraim Moscovich – CA Technologies
- Bryan Murray – Hewlett-Packard Company
- Steven Neely – Cisco
- Ryuichi Ogawa – NEC Corporation
- John Parchem– Microsoft Corporation
- Shishir Pardikar – Citrix Systems Inc.
- Miguel Peñalvo – Telefónica
- Gilbert Pilz – Oracle
- Alvaro Polo – Telefónica
- Enrico Ronco – Telecom Italia
- Federico Rossini – Telecom Italia
- Matthew Rutkowski – IBM
- Tom Rutt – Fujitsu
- Hemal Shah – Broadcom
- Nihar Shah – Microsoft Corporation
- Alan Sill – Texas Tech University
- Zhexuan Song – Huawei
- Marvin Waschke – CA Technologies
- Eric Wells – Hitachi, Ltd.
- Jeff Wheeler – Huawei
- Maarten Wiggers – Fujitsu
- Daniel Wilson – Ericsson AB
- Steve Winkler – SAP AG
- Jack Yu – Oracle
- Aaron Zhang – Huawei
- HengLiang Zhang – Huawei

Cloud Infrastructure Management Interface (CIMI) Model and RESTful HTTP-based Protocol

1 Scope

This specification describes the model and protocol for management interactions between a cloud Infrastructure as a Service (IaaS) Provider and the Consumers of an IaaS service. The basic resources of IaaS (machines, storage, and networks) are modeled with the goal of providing Consumer management access to an implementation of IaaS and facilitating portability between cloud implementations that support the specification. This document specifies a Representational State Transfer (REST)-style protocol using HTTP. However, the underlying model is not specific to HTTP, and it is possible to map it to other protocols as well.

CIMI addresses the management of the life cycle of an infrastructure provided by a Provider. CIMI does not extend beyond infrastructure management to the control of the applications and services that the Consumer chooses to run on the infrastructure provided as a service by the Provider. Although CIMI may be to some extent applicable to other cloud service models, such as Platform as a Service (PaaS) or Storage as a Service ("SaaS"), these uses are outside the design goals of CIMI.

1.1 Document structure

This document defines a model and a RESTful HTTP-based protocol.

The core REST patterns are defined first and, after each resource is defined, any HTTP-specific information for that resource is specified.

1.2 Document versioning scheme

This document adheres to the versioning scheme defined in clause 6.3 of DSP4004.

As the specification changes over time certain features might be deprecated. These are identified in the specification and should not be supported. Each of these deprecated features is clearly denoted in the clause in which they were previously defined.

1.3 Typographical conventions

This specification uses the following conventions:

In the narrative text of the specification:

- The regular or narrative font is Arial.
- Proper CIMI nouns such as Resource names, attribute names, operation names, reserved variable names are in `Courier` font. (e.g., `Machine`, `volumes`, `$expand`). The plural form applies to such names to indicate several instances of such Resources (e.g., `Machines, Systems).`
- Example text is in small `Courier` font and over a darker background.
- Quotes are used for any text that needs be distinguished as a name or value of a particular concept (e.g., the "value constraints" attribute, the “Resource Name” column, a “false” value). In such cases, the string in quotes is always qualified by the concept it is an instance of.
- Names for CIMI concepts that may be common English words but have a very specific meaning in CIMI, are in narrative font but capitalized, e.g., Provider, Consumer, Resource, Collection.

When used in their common English sense they remain lowercase. However, CIMI modeling concepts that are used in a commonly understood manner remain in lowercase, such as: attribute, operation.

Inside tables describing the Resource data model:

- The narrative font is used for all terms, as the table structure qualifies them sufficiently.
- Where textual descriptions are introduced, the rules for narrative text apply.
- Names that are used as types (i.e., names of embedded structures as well as atomic types such as "integer", "string"), are in *italic*.
- Names that are just placeholders for actual names that may vary with each model instance, are shown between < > (e.g., <componentTemplate>).

Where the serialization of Resources is described, a pseudo-schema notation is used with the following conventions:

- Values in *italics* indicate data types instead of literal values.
- Characters are appended to items to indicate cardinality:
  - "?" (0 or 1)
  - "*" (0 or more)
  - "+" (1 or more)
- Vertical bars, "|", denote choice. For example, "a|b" means a choice between "a" and "b".
- The characters {, }, [, and] are block delimiters within the pseudo-schema. (Blocks may extend over multiple lines.)
- Parentheses, "(" and ")" are used in the pseudo-schema only to indicate the scope of the operators "?", "*", "+" and "|".
- Ellipses (i.e., "...") indicate points of extensibility. Note that the lack of an ellipses does not mean no extensibility point exists, rather it is just not explicitly called out - usually for the sake of brevity.
- The scope of "?", "*", "+" and "|" follows these rules:
  - If immediately following a block delimiter or an array closing symbol e.g., "], ?" the scope is the entire block.
  - If not following any closing block delimiter, the scope is everything that precedes it on the same single line.

Operation names Create, Update, Delete, Read are abstract operations that convey the semantics of concrete corresponding operations, such as HTTP methods or CIMI operation URIs.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated or versioned references, only the edition cited (including any corrigenda or DMTF update versions) applies. For references without a date or version, the latest published edition of the referenced document (including any corrigenda or DMTF update versions) applies.

DMTF DSP0223, *Generic Operations 1.0*, http://www.dmtf.org/standards/published_documents/DSP0223_1.0.pdf

DMTF DSP0243, *Open Virtualization Format Specification 1.1*, http://www.dmtf.org/sites/default/files/standards/documents/DSP0243_1.1.pdf

DMTF DSP0262, *Cloud Audit Data Federation (CADF) -Data Format and Interface Definitions Specification version 1.0.0*, http://dmtf.org/sites/default/files/standards/documents/DSP0262_1.0.0.pdf

DMTF DSP1001, *Management Profile Specification Usage Guide 1.1,* http://www.dmtf.org/standards/published_documents/DSP1001_1.1.pdf

DMTF DSP4004, *DMTF Release Process 2.4*, http://www.dmtf.org/sites/default/files/standards/documents/DSP4004_2.4.pdf

IANA HTTP Header Registry, http://www.iana.org/assignments/message-headers/perm-headers.html

IEC 80000-13:2008, International Organization for Standardization, Geneva, Switzerland, *Quantities and units – Part 13: Information science and technology*, April 2008, http://www.iso.org/iso/catalogue_detail?csnumber=31898

IEEE 802.3-2012, IEEE Standards Association. *IEEE Standard for Ethernet*, December 2012, http://standards.ieee.org/findstds/standard/802.3-2012.html

IETF RFC791, Postel, J.,*Internet Protocol*, September 1981, http://www.ietf.org/rfc/rfc791.txt

IETF RFC2460, Deering, S. and R. Hinden, *Internet Protocol, Version 6 (IPv6) Specification*, December 1998, http://www.ietf.org/rfc/rfc2460.txt

IETF RFC7230, R. Fielding et al, *Hypertext Transfer Protocol -(HTTP/1.1)*, http://www.ietf.org/rfc/rfc7230.txt

IETF RFC3986, T.Berners-Lee et al, *Uniform Resource Identifiers (URI): Generic Syntax*, August 1998, http://www.ietf.org/rfc/rfc3986.txt

IETF RFC4291, Deering, S. and R. Hinden, *IP Version 6 Addressing Architecture*, February 2006, http://www.ietf.org/rfc/rfc4291.txt

IETF RFC4627, D. Crockford, *The application/json Media Type for JavaScript Object Notation (JSON)*, July 2006, http://www.ietf.org/rfc/rfc4627.txt

IETF RFC5246, T. Dierks and E. Rescorla, *The Transport Layer Security (TLS) Protocol Version 1.2*, http://www.ietf.org/rfc/rfc5246.txt

ISO 8601:20044, International Organization for Standardization, Geneva, Switzerland, *Data elements and interchange formats -- Information interchange - - Representation of dates and times*, March 2008, http://www.iso.org/iso/iso_catalogue/ catalogue_tc/catalogue_detail.htm?csnumber=40874

ISO/IEC 14977:1996, Roger S. Scowen, *Extended BNF — A generic base standard.* Software Engineering Standards Symposium 1993. http://www.iso.org/iso/catalogue_detail?csnumber=26153

ISO/IEC Directives, Part 2, *Rules for the structure and drafting of International Standards*, http://isotc.iso.org/livelink/livelink.exe?func=ll&objId=4230456&objAction=browse&sort=subtype

NIST Special Publication 800-145, Peter Mell and Timothy Grance, *The NIST Definition of Cloud Computing*, Sept. 2011, http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

NIST Special Publication 500-292, Fang Liu, Jin Tong, Jian Mao, Robert Bohn, John Messina, Lee Badger and Dawn Leaf, *NIST Cloud Computing Reference Architecture*, Sept. 2011, http://collaborate.nist.gov/twiki-cloud-computing/pub/CloudComputing/ReferenceArchitectureTaxonomy/NIST_SP_500-292_-_090611.pdf

Representational State Transfer, Roy Fielding, Doctoral dissertation, University of California, *Architectural Styles and the Design of Network-based Software Architectures (Chapter 5)*, 2000, http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

Unicode Standard, Unicode Consortium, *The Unicode Standard*, Version 2.0, Addison-Wesley, 1996.

XMLSchema - Part 1, World Wide Web Consortium (W3C) Recommendation, H. Thompson, et al., Editors, *XML Schema Part 1: Structures Second Edition*, 28 October 2004, http://www.w3.org/TR/xmlschema-1/

XMLSchema - Part 2, World Wide Web Consortium (W3C) Recommendation, P. Biron, A. Malhotra, Editors, *XML Schema Part 2: Datatypes (Second Edition)*, 28 October 2004, http://www.w3.org/TR/xmlschema-2/

3 Terms and definitions

In this document, some terms have a specific meaning beyond the normal English meaning. Those terms are defined in this clause.

The terms "shall" ("required"), "shall not," "should" ("recommended"), "should not" ("not recommended"), "may," "need not" ("not required"), "can" and "cannot" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Annex H. The terms in parenthesis are alternatives for the preceding term, for use in exceptional cases when the preceding term cannot be used for linguistic reasons. Note that ISO/IEC Directives, Part 2, Annex H specifies additional alternatives. Occurrences of such additional alternatives shall be interpreted in their normal English meaning.

The terms "clause," "subclause," "paragraph," and "annex" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Clause 5.

The terms "normative" and "informative" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Clause 3. In this document, clauses, subclauses, or annexes labeled "(informative)" do not contain normative content. Notes and examples are always informative elements.

The terms defined in DSP4004, DSP0223, and DSP1001 apply to this document. The following additional terms are used in this document.

3.1 authentication

The process of verifying a claim, made by a subject, that it should be allowed to act on behalf of a given principal (person, service, etc.). Typical authentication mechanisms involve the use of username/password combination or public/private key pairs.

3.2 authorization

The process of verifying that an authenticated principal (person, service, etc.) has permission to perform certain operations (e.g., read, update) on specific Resources. (Also known as Access Control.)

3.3 cloud
Synonymous with “cloud computing” as defined in section 2 of the NIST Definition of Cloud Computing [SP800-145].

3.4 Cloud Service Consumer
A category of actors that includes the Consumer Business Manager (who approves business and financial expenditures for consumed services; accounts for used service instances; establishes business relationships; sets up accounts, budget, and terms; etc.); the Consumer Service Administrator (who requests service instances and changes to service instances; purchases services within the business relationship; creates Service Users (including policies); allocates resources, such as computer and storage; generates reports, such as usage; etc.); and Service Users (who use service instances provided by a Cloud Service Provider). The term "Consumer" is used if the indicated action or activity could involve one or more of the above actors. In cases where the distinction between the actors in this category is relevant, the more detailed term is used.

For purposes of comparison and alignment, it should be noted that a Cloud Service Consumer is equivalent to the “Cloud Consumer” actor defined in the NIST Reference Architecture [SP500-292].

3.5 Cloud Service Provider
A category of actors that includes the Service Operations Manager (who manages the technical infrastructure required for providing cloud services; monitors and measures performance and utilization against SLAs; provides reports from monitoring and measurement; etc.); Service Business Manager (who offers all types of services developed by cloud service developers; accounts for services potentially offered by service Providers themselves and services offered on behalf of cloud service developers; establishes a portfolio of business relationships; and sets up accounts and terms for Consumers, etc.); and Service Transition Manager (who enables a customer to use the cloud service, including "onboarding", integration, and process adoption; defines and creates service offerings based on Templates and Configurations that can be used by Consumers and are populated into the catalog; etc.). The term "Provider" is used if the indicated action or activity could involve one or more of the above actors. In cases where the distinction between the actors in the category is relevant, the more detailed term is used.

For purposes of comparison and alignment, it should be noted that a Cloud Service Provider is equivalent to the “Cloud Provider” actor defined in the NIST Reference Architecture [SP500-292].

3.6 Collection
A particular kind of Resource that contains a collection of other Resources and has a representation and serialization defined in this specification. Synonym for “CIMI collection”.

3.7 Configuration
A set of metadata, the values of which serve as the parameters of a discrete conformation of a specific type of virtual resource.

3.8 Endpoint
An element within a Network Segment from which commincation can originate or to which communication can be sent. Endpoints have a unique, protocol specific, address within a Segment by which they are distinguished.

3.9 Infrastructure as a Service (IaaS)
A cloud computing service model defined in section 2 of the NIST Definition of Cloud Computing [SP800-145].

3.10 Interface
An abstract element of virtual hardware that enables connection to a Network via Endpoints.

3.11 message confidentiality

A quality of a message that prevents anyone but the intended receiver(s) from viewing its contents.

3.12 message integrity

A quality of a message that allows a receiver of that message to determine whether the contents of the message have been altered since its creation.

3.13 Network

A construct that supports communications between elements within a Cloud using one or more protocol specific Segments that support addressable Endpoints.

3.14 Resource

A representation of an entity managed by the [Cloud Service] Provider that is generally available to the [Cloud Service] Consumer to access or operate on by way of the interface described in this specification. Synonym for “CIMI resource”.

3.15 Segment

A component of a Network that supports communication between Endpoints using a single protocol. Also referred to as a Protocol Segment to emphasize that Segments are always bound to a single communication protocol.

3.16 Template

A component Synonym for “CIMI template”. A Resource that represents the set of metadata and instructions used to instantiate some other Resource (e.g., a `MachineTemplate` is used to create `Machines`.

4 HTTP-based protocol

4.1 Introduction

All operations are based on the *HyperText Transfer Protocol (HTTP)*, version 1.1 [RFC2616]. Each request is sent by using an HTTP verb such as PUT, GET, DELETE, HEAD, or POST and includes a message body in either JSON or XML format. Each response uses a standard HTTP status code, whose semantics are interpreted in the context of the particular request that was made. Each Resource in the model has a MIME type that further contextualizes the payload of requests and responses.

Resources in the model are identified by URIs, and each Resource's representation shall contain an "ID" attribute, of type URI, that acts as a "self pointer.” This URI shall be unique within the context of the Provider's implementation. Dereferencing (through an HTTP GET) the URI of a Resource yields a representation of the Resource containing attributes and links to associated Resources. To begin operations, a client shall know the URI to the main entry point of a Provider - also known as the "Cloud Entry Point" Resource. All other Resources within the environment shall then be discoverable by the way of the iterative following of links to associated Resources within each Resource retrieved.

4.1.1 Protocol evolution and client expectations

Future versions of this specification structure changes in such a way that clients that conform to an earlier version of this specification continue to work, and are not be adversely affected by the evolution of the protocol. Clients are expected to follow a few simple rules to ensure this compatibility:

1. Clients shall not assume that the serializations shown for responses in this specification are complete. In particular, clients shall accept responses that contain data mixed in with the serializations shown here, and shall ignore such data. However, per clause 4.2.1.3, clients shall include unknown data in PUT requests to update Resources.

2. Clients shall not assume anything about the operations supported by a server. They are expected to discover operations that are supported (and permissible) by navigating to Resources from the cloud entry point. The serializations of Resources encountered indicate which operations are supported by the server.

4.1.2 XML namespaces

Error! Reference source not found. lists the XML namespaces that are used in this specification. The choice of any namespace prefix is arbitrary and not semantically significant.

Table 1 – XML namespaces

Prefix	XML Namespaces	Specification
cimi	http://schemas.dmtf.org/cimi/2	This specification
xs	http://www.w3.org/2001/XMLSchema	XML Schema Part2

4.1.3 URI space

While URIs returned by Providers are to be treated as opaque by Consumers, and Consumers shall not make assumptions about the layout of the URIs or the structures of the URIs for the Resources, a Consumer may augment URIs with any well-defined query parameters that are supported by the Provider as defined in clause 4.1.6.

The sample URIs used in this specification are not normative and the patterns used shall not be interpreted as guidance for implementations. For example, any of the following URIs might be used by Providers to reference a particular `Machine` Resource:

```
http://example.com/machines/12345
http://example.com/machines?id=12345
http://example.com/12345
http://example.com/Cloud/resource?id=12345
```

4.1.4 Media types

In this specification, Resource and response representations are encoded either in JSON, as specified in RFC4627 or in XML. If serialized in JSON, the media-type for CIMI resources shall be "application/json". If serialized in XML, the media-type shall be "application/xml".

In the JSON serialization of CIMI representations sent by Providers, there shall be an additional attribute on the root object called "`resourceURI`" that contains the unique URI that is associated with the type of CIMI resource being serialized.

Note that this requirement applies even if the `$select` attribute is used to subset the Resource being acted upon.

In the XML serialization of Collection representations sent by Providers there shall be a `resourceURI` attribute, as shown in the example XML serialization of Collections in clause 5.5.12.

This attribute is optional for Consumers to include. If included, this attribute's value shall match the "`typeURI`" attribute of the corresponding `ResourceMetadata` Resource (see clause 5.8), if `ResourceMetadata` is supported. This value shall also be equivalent to the wrapping element of the XML serialization; in other words, the namespace of the wrapper element concatenated a "/" and then its localName.

Any CIMI resource implemented by a Provider shall have representations in JSON and XML. The client implementation may thus use either JSON or XML in requests with any server implementation, and may request a specific serialization using server-driven content negotiation (using the Accept request header).

4.1.5 Request headers

This specification uses general-header, request-header, and entity-header headers as defined in RFC2616 in request messages to provide metadata about the message. Applications using messages defined in this specification shall use headers consistent with the requirements of RFC2616.

4.1.6 Request query parameters

Providers may choose to include query parameters as part of the URIs returned to Consumers. Consumers shall include those query parameters when sending messages to those URIs. CIMI defined query parameters are prefixed with a dollar sign ("$"). If Providers choose to define query parameters, they shall not be prefixed with a dollar sign to avoid conflicts with current and future CIMI defined query parameters.

To modify the behavior of the Provider when processing request messages, Consumers may augment request URIs as described in the following clauses. As stated in clause 4.1.3, URIs returned from Providers are to be treated as opaque by Consumers; however, it is the responsibility of the Consumer to understand the use of the query parameters defined in the following clauses and ensure correctness when making a request.

Unsupported, or unknown, query parameters shall be silently ignored by Providers. Consumers may examine the CloudEntryPoint's capabilities to determine whether support of these query parameters is enabled.

4.1.6.1 Filtering Collections

If retrieving the representation of a Collection, Consumers may include the `$filter` query parameter to reduce the number of entries of the Collection that are returned based on the data within the entries of the Collection. Providers shall interpret and process the `$filter` query parameter as described in this section. The `$filter` parameter shall be of the form:

```
?$filter=expression
```

where "expression" represents a mathematical expression denoting how the top-level attributes of the Resources within the Collection shall be filtered. The expression is defined by the following EBNF grammar:

```
Filter      ::= AndExpr ( 'or' Filter )* ;
AndExpr     ::= Comp ( 'and' AndExpr )*
Comp        ::= Attribute Op Value
              | Value Op Attribute
              | PropExpr
              | '(' Filter ')'
Op          ::= '<' | '<=' | '=' | '>=' | '>' | '!='
Attribute   ::= ? resource attribute name ?
Value       ::= IntValue | DateValue | StringValue | BoolValue
IntValue    ::= /[0-9]+/
DateValue   ::= ? as defined by XML Schema ?
StringValue ::= "..." | '...'
```

```
BoolValue   ::= 'true' | 'false'
PropExpr    ::= 'property[' StringValue ']' Op StringValue
```

Where `PropExpr` is used to find Resources that contain a property with a certain key/value combination. The key is the `StringValue` within the square brackets ([ ]) and the value is the `StringValue` after the `Op`. The Resource shall be considered to satisfy the search criteria if any of the properties in the Resources match the specified `PropExpr`.

Each of these shall be percent encoded in the URL as appropriate.

The choice of which operator (including 'and' and 'or') is limited based on the type of the value and attribute. The following example describes the allowable operators:

```
'or', 'and'                        : Boolean value/attribute
'<', '<=', '=', '>=', ">', '!='    : Integer and date value/attribute
'=', '!='                          : String value/attribute
```

Consumers may include multiple filters within a single URI. Providers shall treat multiple filters as a series of "and" expressions where an entry of the Collection shall only be included in the response message if it satisfies all of the filter expressions specified.

Examples:

In the following examples, the following sample base URIs are used.

The URI to the `MachineCollection` of the Cloud Entry Point is as follows:

```
/machines
```

The URI to a `Machine` is as follows:

```
/machines/123
```

The URI to the `DiskCollection` of a `Machine` is as follows:

```
/machines/123/disks
```

The URI to the `VolumeCollection` of a `Machine` is as follows:

```
/machines/123/volumes
```

To filter the `MachineCollection` so that just `Machines` with a "name" attribute of "mine" are returned, use the following filter:

```
GET /machines?$filter=name='mine'
```

To filter a `DiskCollection` of a `Machine` so that just `Disks` with a format of "ntfs" are returned, the following filter would be used:

```
GET /machines/123/disks?$filter=format='ntfs'
```

If the `$filter` parameter is used, the Collection's "`count`" attribute shall contain the number of Resources matching the filter expression.

4.1.6.2 Subsetting Collections

If retrieving the representation of a Collection, Consumers may include query parameters to subset the number of entities of the Collection that are returned. Providers shall interpret and process these query parameters as described in this clause. While the previous clause discussed how to perform a filter over the data within the Collection, this clause uses ordinal position within the Collection to achieve the desired reduction.

This specification defined two query parameters that, if used, shall indicate the first and last ordinal positions of the entities within the Collection that are returned. The query parameters shall be of the form:

```
?$first=number
?$last=number
```

Where "`$first`" indicates the (1-based) ordinal position of the first entity of the Collection to return and "`$last`" indicates the (1-based) ordinal position of the last entity of the Collection to return. Consumers are not required to use both at the same time. If `$first` is specified but `$last` is not, the implied value for `$last` shall be the ordinal position of the last entity in the Collection. Conversely, if `$last` is specified but `$first` is not, the implied value for `$first` shall be 1.

If Consumers include these query parameters, the ordinal positions of entries in the collection before subsetting shall be stable when no changes are made to the collection or its entries. If filtering or sorting are used in the same query, the subsetting applies to the collection resulting from those operations.

If any part of the range as expressed by `$first` and `$last` is outside of the bounds of the Collection, just the Resources (if any) in the Collection that are contained within that range shall be returned. A fault shall not be generated if any part, or all, of the expressed range is outside the bounds of the Collection. Note that if `$first` is larger than `$last`, the range shall represent an empty range and therefore no Resources are returned.

If either `$first` or `$last` are specified, and a filter expression (as defined in clause 4.1.6.1) is also specified, the filter expression shall be performed first and then the ordinal constraints of `$first` and `$last` shall be applied.

The inclusion of $first or $last does not affect the value of the Collection's returned "count" attribute: it shall contain the number of Resources in the Collection before subsetting. In case filtering is also used, “count” shall be the size of the Collection resulting from the filtering.

4.1.6.3 Subsetting Resources

If retrieving the representation of a Resource, Consumers may include the `$select` query parameter to specify a subset of the Resource to be acted upon. Providers shall interpret and process this query parameter as described in this section. This subsetting shall have the semantic equivalence of referencing a different Resource whose attributes are a subset of the original Resource as specified by the attribute names listed in the `$select` query parameter. The format of a `$select` query parameter is:

```
?$select=attributeName,...
```

The value of the `$select` query parameter shall be a comma-separated list of top-level attribute names of the Resource, possibly including the string “operations” in case the intent is to select the operations available to the Consumer for this Resource. Any attribute name erroneously appearing in the list that is not part of the Resource shall be ignored by the Provider. An attribute name of "*" is equivalent to specifying all of the attributes of the Resource including its operations. Any attribute name explicitly appearing more than once in a URI shall have its second (and subsequent) appearances ignored.

The `$select` query parameter may appear more than once in a URI. This is semantically equivalent to all of the attribute names appearing as values of a single `$select` query parameter. For example:

```
?$select=name&$select=state
```

is equivalent to:

```
?$select=name,state
```

The order of attribute names in the `$select` query parameter is not relevant for serialization purposes. The attributes are serialized per the serialization rules/order as specified by the Resource definition.

Note that per clause 4.1.4, if a Resource representation is sent by a Provider it shall always include the `resourceURI` attribute even if it is not specified in the `$select` query parameter.

For example, to subset the list of Machine attributes being acted upon to just the "name" and "description", the following query parameter would be used:

```
?$select=name,description
```

See clause 4.2.1.3.1 for more information about the impact of using this query parameter when updating a Resource.

If `$select` is used in the URI for a Collection resource, the subsettings shall apply to the attributes of the Collection resource itself as for any other Resource. For example, to subset a Collection resource in order to only return the number of its items, plus the operations available on this Collection:

```
?$select=count,operations
```

However, exceptionally for Collection resources, if some attribute provided in the `$select` list is not a top-level attribute of the Collection resource but instead is an attribute of the entities that are items of the Collection, the subsetting shall apply to each item of the Collection regarding this attribute. For example, if retrieving the `DiskCollection`, the following query parameter:

```
?$select=name,capacity
```

returns a collection of the `Disks` associated with a `Machine` but each entity of the collection just has the `name` and `capacity` attributes and nothing else, not even the `operations` or `id` attributes.

Optionally, an implementation may also support the alternative attribute name notation: <`collectionName`>/<`attributeName`> for subsetting the items inside a collection. For example, the following subsetting on items of a `Disks` Collection is equivalent to the one done in the previous example, while in addition listing the operations of the Collection resource itself (not of its items):

```
?$select=disks/name,disks/capacity,operations
```

This notation, if supported (see the “QueryPathNotation” capability in 5.11.1), allows for disambiguating subsettings if the same attribute name can be found for the Collection and for each item in the collection (which is always the case for `id` and `operations`).

4.1.6.4 Expanding references

If retrieving the representation of a Resource, Consumers may include the `$expand` query parameter to specify which of the top-level "`reference`" attributes of the Resource shall be "`expanded`". Providers shall interpret and process this query parameter as described in this clause. To expand a reference means that the attributes of the Resource being referenced shall be included in the serialization of that attribute. This feature allows for a more optimized retrieval of Resources.

The serialization shall be performed as follows:

JSON serialization:

```
"name": { "href": string }
```

shall be expanded to be:

```
"name": {
  "href": string,
  ... attributes of referenced resource...
}
```

XML serialization:

```
<name href="xs:anyURI"/>
```

shall be expanded to be:

```
<name href="xs:anyURI">
  ... attributes of the referenced resource...
</name>
```

Note that in the XML case the nested elements shall not contain the wrapper element of the referenced Resource (e.g., <`Machine`> in the case of a reference to a `Machine` Resource).

The format of a `$expand` query parameter shall be:

```
?$expand=attributeName,...
```

The value of the `$expand` query parameter is a comma-separated list of attribute names. Any attribute name erroneously appearing in the list that is not part of the Resource, or is not a reference, shall be ignored by the Provider. An attribute name of "*", or no attribute name list at all, is equivalent to specifying all of the attributes. Any attribute name explicitly appearing more than once in a URI shall have its second (and subsequent) appearances ignored.

The `$expand` query parameter may appear more than once in a URI, which is semantically equivalent to all of the attribute names appearing as values of a single `$expand` query parameter.

If the Resource being retrieved is a Collection, the attribute names listed in the `$expand` shall apply to the attributes of the entities within the Collection. For example, specifying:

```
?$expand=volumes
```

if retrieving the `MachineCollection` has the same net effect as applying the "`expand`" semantics to the specified attribute ("`volumes`" in this example) of each `Machine` within the Collection. To be clear, `$expand` acts on the attributes of the Resources in the Collection, not on the wrapping Collection Resource itself.

4.1.6.5 Specifying the Resource format

If retrieving the representation of a Resource, the HTTP Accept header is used to specify the encoding style of the response. While it is recommended that Consumers use the Accept header, there might be situations where Consumers are unable to control the values specified in that header. In these cases Consumers may use the `$format` query parameter to override the Accept header values. Providers shall interpret and process the `$format` query parameter as described in this clause.

The `$format` parameter shall be of the form:

```
?$format=encoding
```

Where "`encoding`" is the requested representation of the response. This specification defines two possible values: "`json`" and "`xml`". Providers may support others. The value of the `$format` query parameter shall be case insensitive.

If both an Accept header and `$format` query parameter are present in a request message, the `$format` value shall take precedence. If the `$format` query parameter appears more than once, the second, and subsequent, appearances shall be ignored.

4.1.6.6 Sorting Collections

If retrieving the representation of a Collection, Consumers may include the `$orderby` query parameter to sort the entries of the Collection that are returned based on different attributes or in a different order (descending). Providers shall interpret and process the `$orderby` query parameter as described in this section. The `$orderby` parameter shall be of the form:

```
?$orderby=attributeName[:asc|:desc], …
```

The `$orderby` expression may include multiple, comma-separated attribute names. Each attribute name may be optionally followed immediately by a colon and "`asc`" to denote ascending order (default), or "`desc`" to denote descending order for that attribute. If neither `asc` nor `desc` is specified, the order shall be "`ascending`".

The attributes included in the `$orderby` shall be of the following types as defined in clause 5.5: boolean, dateTime, duration, integer, or string.

The sort shall be performed based on the attribute type.

The following rules apply to the ascending sort order:

- boolean – 'false' shall come before 'true'.
- dateTime – An earlier datetime shall come before a later datetime.
- duration – A shorter duration shall come before a longer duration.
- integer – Smaller integers shall come before larger integers. Negative integers shall come before positive integers.
- string – Ordering is based on a binary comparison of the transformed strings according to the rules of the Normalization Form KD of the Unicode standard as defined in Unicode Standard Annex (UAX), annex #15 .

For the `desc` sort order, the reverse of the above shall be performed.

Examples:

To sort the result set of the `MachinesCollection` Resource on the "`created`" attribute in descending order, the following expression would be used:

```
GET /machines?$orderby=created:desc
```

To sort the result set of the `MachinesCollection` Resource on the "`cpu`" attribute in descending order, followed by the "`memory`" attribute in ascending order, the following expression would be used:

```
GET /machines?$orderby=cpu:desc,memory:asc
```

If collection subsetting is used in the same query, the subsetting applies to the sorted collection. When no `$orderby` is specified, the order of entries in the returned Collection is not defined.

4.1.7 Response headers

As defined in RFC2616, this specification uses general-header, response-header, and entity-header headers in response messages to provide metadata about the message. Applications that use messages defined in this specification shall use headers consistent with the IANA HTTP Header Registry.

4.1.7.1 Job header

If the server supports the `Job` Resource, response messages shall include a header defined by this specification to indicate the URI for the job created to process the associated request message.

```
CIMI-Job-URI = "CIMI-Job-URI" ":" string
```

4.1.7.2 ETag support

An ETag header may be provided by a Provider with each Resource as specified in RFC2616. If a Provider does provide an ETag header, it shall also support If-Match header processing on behalf of the Consumer.

4.2 Protocol operations

This clause defines the set of common HTTP operations that a Provider may expose. At its core, there are four basic CRUD (Create, Read, Update, and Delete) operations. The manner in which these are used is consistent across all Resources within the model; therefore, their use is defined once and is to be applied consistently. Some Resources support specialized operations that do not fit well into a CRUD style of operation and those follow a similar high-level pattern, but each operation is allowed to have slight variations to accommodate its specific needs. The specifics of these special operations are detailed within the clause that defines the Resource.

If appropriate, some of the Resource representations include an "operations" attribute. Providers shall only include the "operations" attribute if the specified operations are accessible to the current client for that particular Resource. This situation means that based on many factors (e.g., authorization rights of the clients, current state of the Resource, etc.), a different set of "operations" shall be returned on each serialization of the Resource.

Each operation shall include a "rel" and an "href" field. The "rel" field shall uniquely identify the operation name (e.g., "add", "edit"), while the "href" field is the URI to which the operation's request message shall be sent. Note that the "href" field's URI may be different from the URI of the Resource itself. Each operation may have an “available” field to indicate that the operation can be performed by the Consumer. The "available" field is of type boolean with a default value of "true". If “available” is set to “false” it indicates that the operation is not currently available. This would normally indicate a temporary condition. For example, some Machine operations may not be available depending on the state of the Machine.

The operations attribute shall be serialized as follows:

JSON serialization:

```
{ "operations": [
    { "rel": string, "href": string, (“available”: boolean)? }, +
  ]
}
```

XML serialization:

```
<operations xmlns="http://schemas.dmtf.org/cimi/2">
  <operation rel="xs:anyURI" href="xs:anyURI" (available=”xs:boolean”)? /> *
</operations>
```

For example, the "edit" operation would appear as:

JSON serialization:

```
{ "operations": [
    { "rel": "edit", "href": "<editURI>" }
```

```
  ]
}
```

XML serialization:

```
<operations xmlns="http://schemas.dmtf.org/cimi/2">
  <operation rel="edit" href="<editURI>"/>
</operations>
```

Additional "`rel`" values may be defined by Providers; however, they shall be fully qualified URIs and not relative URIs.

4.2.1 Common CRUD operations

Each of the Resources supported by this protocol shall adhere to the interaction patterns defined in the following clauses.

4.2.1.1 Creating a new Resource

To create a new instance of a Resource type, an HTTP POST request is sent to a designated "`addURI`" for that Resource type. In many cases, the Collection resource that maintains, or groups, all instances of that Resource type includes an "add" operation. The "add" operation references the `addURI` that is to be used.

The HTTP POST request shall include:

- CIMI serialization of the request to create a new Resource in the HTTP Body
- HTTP Content-Type header
- HTTP Content-Length header

For example, the request can be:

```
POST <addURI> HTTP/1.1
Host: <hostname>
Accept: application/(json|xml)
Content-Type: application/(json|xml)
Content-Length: <length>

<serialization of request to create a new resource>
```

This example has an Accept header with one of the CIMI supported media types: application/json or application/xml. If the Provider chooses to reply with a serialization, this serialization should be of the specified media type. Omission of the Accept header allows the Provider to reply with a serialization of any media type. If the Resource has a “`State`” attribute, its value shall be “CREATING” while the Provider is processing this operation.

Many of the create requests are defined such that a Template of the new Resource is passed. These create requests allow for the Template to be passed in "by-reference" or "by-value." For example, creating a new `Machine` looks like this (here using XML):

```
<MachineCreate xmlns="http://schemas.dmtf.org/cimi/2">
  <name> xs:string </name> ?
  <description> xs:string </description> ?
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties>
<machineTemplate href="xs:anyURI"? >
    ... template attributes ... ?
  </machineTemplate>
</MachineCreate>
```

Note that in the XML case the creation of a new `Machine` requires a wrapper element named `MachineCreate` per the rules specified in clause 5.5.12.1.

More generally, creating a new Resource shall follow one of these two serialization patterns (here illustrated in JSON):

(1) Resource creation by passing a Template by value:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/ResourceCreate",
  "name": "myResourceName", ?
  "description": "My resource description", ?
  "properties": { "prop1name" : "prop1value" , + }, ?
  "resourceTemplate": {
    <here the template is passed by value>
  }
}
```

Where *resourceTemplate* is the actual name of the template for that Resource.

(2) Resource creation by passing a template by reference:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/ResourceCreate ",
  "name": "myResourceName", ?
  "description": "My resource description", ?
  "properties": { "prop1name" : "prop1value" , + }, ?
  "resourceTemplate": { "href": string ,
    <here some template attribute/value pairs may be added to override values in the
referenced template>
  }
}
```

In case the created Resource is itself a Template, only the first creation pattern - by value - applies.

In both patterns (1) and (2) the `resourceURI` attribute specifies the operation here generically identified as "ResourceCreate", e.g., `MachineCreate`.

In both patterns (1) and (2) an element corresponding to the Resource Template (here identified generically as "resourceTemplate" e.g., `MachineTemplate`) is specifying the Template to be used, either by value (1) or by reference (2).

Direct setting of attributes in the new Resource:

In a creation request it is possible to set the value of some attributes of the newly created Resource, regardless of what values the Template instantiation might have set if used alone. Three common attributes of the newly created Resource may be set: `name`, `description`, and `properties`.

The semantics shall be same as of a partial update of the Resource for these attributes (described in a next subclause), immediately following the Resource creation from the Template alone.

Defining or referring to the Resource Template:

In pattern (1) above, the Provider may choose to create a Template Resource from the value given, but such creation is temporary in nature. The Provider shall not expose such a transient Resource to the Consumer and no such transient Resource shall be included in any query results back to the Consumer.

In pattern (2) above, additional attribute name/value pairs may be given inside the ResourceTemplate element that also contains the reference to the external (pre-existing) Template in order to override similar attributes defined in the Template. More precisely:

- Any top-level attribute of complex or simple type in the referred Template shall be overridden by providing its name/value pair in the create request inside the resourceTemplate element and immediately under it. For a top-level attribute of a complex type (e.g., arrays, Collections, structures), the provided complex value shall also set all underlying attributes – e.g., array elements.
- The semantics shall be same as of modifying (overriding) parts of the referred Template just before it is used for instantiation, but these overrides shall not persist in the referred Template and shall only concern this particular instantiation.

In pattern (2) above, Consumers may erase any Template attributes by specifying either

```
"attribute": null
```

for the attribute in the JSON serialization, or

```
<attribute/>
```

in the XML serialization for that attribute.

Some of the create requests allow for configuration type of Resources to be passed by-reference or by-value as well - e.g., `Credential` on a `Machine` create operation. The processing rules defined above applies in those cases as well.

If the response has a 201 status code, the response shall include:

- HTTP Location header with a reference to the new Resource

If the response to a create request includes a serialization of the new Resource, the response shall additionally include:

- HTTP Content-Type header
- HTTP Content-Length header

For example, the response can be:

```
HTTP/1.1 201 Created
Location: <location>
Content-Type: application/(json|xml)
Content-Length: <length>

<serialization of new resource>
```

4.2.1.2 Retrieving a representation of a Resource

To retrieve a representation of Resource, an HTTP GET request is sent to the Resource's URI.

For example, the request can be:

```
GET <ResourceURI> HTTP/1.1
```

```
Host: <hostname>
Accept: application/(json|xml)
```

If the response has a 200 status code, the response shall include:

- HTTP Content-Type header
- HTTP Content-Length header

For example, the response can be:

```
HTTP/1.1 200 OK
Content-Type: application/(json|xml)
Content-Length: <length>

<serialization of resource>
```

4.2.1.3 Updating a Resource

To update a Resource's state, an HTTP PUT request containing the complete, updated representation is sent to a designated `editURI` for that Resource type. Consumers shall include all non-empty attributes of the Resource in the PUT request - including ones that it might not support or understand that were returned in a GET response. This is to ensure that a client does not inadvertently modify (erase) data in a Resource by excluding it from the full representation of the Resource.

In many cases, this `editURI` is the same as the URI of Resource itself. Retrieving the Resource representation shall include an "edit" operation, which contains the `editURI` that is to be used, if the requester is allowed to modify the Resource.

While processing a PUT request, if the server detects that an attempt is being made to update a read-only, or immutable, attribute, it shall silently ignore that attribute update request and shall not generate an error. This rule applies to Resource partial updates as well.

Because of potential conflicts that might occur due to multiple concurrent updates, Consumers should use the partial update mechanism, defined in 4.2.1.3.1, to reduce the chances of mistakenly updating attributes with out-of-date data.

The HTTP PUT request shall include:

- CIMI serialization of the updated Resource in the HTTP Body
- HTTP Content-Type header
- HTTP Content-Length header

For example, the request can be:

```
PUT <editURI> HTTP/1.1
Host: <hostname>
Accept: application/(json|xml)
Content-Type: application/(json|xml)
Content-Length: <length>

<serialization of request to update a resource>
```

If the response includes a serialization of the updated Resource and has a status code of 200, this response shall include:

- HTTP Content-Type header
- HTTP Content-Length header

For example, the response can be:

```
HTTP/1.1 200 OK
Content-Type: application/(json|xml)
Content-Length: <length>

<serialization of updated resource>
```

4.2.1.3.1 Partial updates to a Resource

For clarity, this clause explains how to use the `$select` query parameter (see clause 4.1.6.3) to subset a Resource for the purposes of only operating on a selected set of top-level attributes.

To update only certain top-level attributes of a Resource, a Consumer may include only the altered attributes in the representation of the Resource within the HTTP request body. If this request is made, the URI to the Resource shall include the attributes to be modified as a comma-separated list of query parameters; in other words, the URI shall be of the form:

```
http://example.com/resource?$select=attribute1,attribute2,...
```

Only the attributes listed in the URI's query parameters shall be modified; attributes not listed in the URI shall not be directly modified by the request. Note that this circumstance does not preclude the modification of one attribute causing side-effects that result in the modification of an attribute not listed in the query parameters.

Any attribute listed in the URI but not included within the HTTP request body shall be reset to a Resource specific value (e.g., removed).

From an HTTP perspective, the updated subsetted Resource is a distinct one. The semantics of a normal HTTP PUT are adhered to; it is a complete replacement update of the specified Resource. From the Consumer’s perspective, the partial update is interpreted and executed by the Cloud Service Provider, and some part of the Resource is changed.

Adhering to the generic PUT semantics defined previously, any attribute of the original (full) Resource included within the HTTP request body shall result in an error being generated if that attribute is not listed in the `$select` query parameter - see clause 5.3. Note that this is due to these attributes being unknown to this subsetted Resource.

The following sample request updates just the name and description attributes of a `Machine`:

```
PUT /machines/myMachine?$select=name,description HTTP/1.1
Host: <hostname>
Accept: application/xml
Content-Type: application/xml
Content-Length: <length>
<Machine>
  <name>My New Machine</name>
</Machine>
```

The `name` attribute is set to "`My New Machine`" and the `description` attribute is erased.

4.2.1.4 Deleting a Resource

To delete a Resource, an HTTP DELETE request is sent to a designated `deleteURI` for that Resource type. In many cases, this `deleteURI` is the same as the URI of Resource itself. Retrieving the Resource representation shall include a "delete" operation, which contains the `deleteURI` that is to be used, if the requester is allowed to delete the Resource.

For example, the request can be:

```
DELETE <deleteURI> HTTP/1.1
Host: <hostname>
```

If the Resource has a `State` attribute, its value shall be “DELETING”, while the Provider is processing this operation.

For example, the response can be:

```
HTTP/1.1 200 OK
```

4.2.1.5 Other operations

While some modifications to the Resources in the model can be done by the way of a simple update (PUT) operation to the Resource's `editURI`, sometimes a more complex set of actions needs to be taken. In these cases, the operations shall be modeled as HTTP POSTs to the operation specific URI of the Resource.

For each of the Resources that define additional operations, a description of the HTTP request and response bodies is provided. However, the general HTTP interaction are as described below.

The request shall be of the following form:

```
POST <operationURI> HTTP/1.1
Host: <hostname>
Accept: application/(json|xml)
Content-Type: application/(json|xml)
Content-Length: <length>

<serialization of request to perform some action>
```

The form of the response varies depending on the operation and is defined by the operation itself.

Note that the definition of the Create operation (see clause 4.2.1.1) follows this same pattern. It is just called out for ease of reference.

4.2.1.6 Synchronous operations

If a Provider supports the `Job` Resource, each incoming PUT, DELETE, POST request shall result in a Job Resource being created and an absolute URI reference to that Job Resource shall be returned back to the client by the way of the CIMI-Job-URI HTTP Header in the HTTP response message:

```
CIMI-Job-URI: <uri-to-Job>
```

In this case, the requested operation shall be complete and the Job URI shall point to a completed Job. If the Job is not complete, the server shall return a 202 and follow the instructions for Asynchronous operations.

4.2.1.7 Asynchronous operations

In some cases, an operation requested by the client may take an undetermined amount of time to be completed. For example, creating a new `Machine` or starting an existing Machine may take a relatively long time to be completed. In these cases, it is not practical to complete these operations within a reasonable HTTP request timeout interval, so the Provider shall return an HTTP "202 Accepted" response code.

As with synchronous operations, if a Provider supports the `Job` Resource, it shall create a `Job` Resource for the incoming request and return a reference to that `Job` Resource back to the client by the way of the CIMI-Job-URI HTTP Header in the HTTP response message. Additionally, in the case of a "202 Accepted" response code, the Provider may also return any of the following in the HTTP response body:

- A representation of the `Job` Resource, if one was created.
- A partial representation of the response message as if the operation were a synchronous operation. For example, when creating a new `Machine`, the response message may include a partial representation of the new `Machine` in the response message. The list of attributes of the Resource that is returned is implementation specific and based upon how much information is available at the time the response message is generated, but it shall be consistent with the definition of the full Resource representation. In the case of a create operation, the Provider may also include an HTTP Location header referencing the "to be created" Resource, if it is known.
- An empty response body.

Note that the decision as to whether any particular operation is synchronous or asynchronous is at the server's discretion.

4.2.2 Error handling

In cases where an error occurs during the processing of a request, the Provider shall include a representation of a `Job` Resource describing the status of the failed operation. This representation of a `Job` shall be included even in cases where the Provider does not expose `Job` Resources. This is to ensure that Consumers are provided with sufficient information, in a consistent manner, as to the reason for the failure. A transient `Job` Resource may be created by the Provider just for error reporting. In case a `Job` Resource is not intended to be used for more than error reporting, the returned "id" attribute shall be an empty path (i.e., "") and the `nestedJobs` array shall be expanded (see 4.1.6.4) to inline the representation of the pseudo subordinate `Jobs`.

4.3 OVF support

The *Open Virtualization Format (OVF) Specification* (DSP0243) describes an open, secure, portable, efficient, and extensible format for the packaging and distribution of software to be run in virtual machines. OVF support in CIMI allows an OVF package to be used to create CIMI management resources by importing the package. Additionally, CIMI management resources can be exported into an OVF package. The actual support for the OVF package is typically provided by a hypervisor that is managed by the CIMI provider. The import of an OVF package exposes CIMI specific constructs and parameters as a result of the import without altering the original OVF package. Thus the CIMI resources that are created as a result of the import form a “View” of what the hypervisor did; however, other (non-CIMI mapped) information from the OVF package may have been used by the hypervisor in its import. This other information is implementation dependent and is not further touched upon by this standard.

An OVF package can support single virtual machines (VMs) corresponding to a single CIMI `Machine` or `MachineTemplate` (see clause 5.14.1) or may also support a complex hierarchy of VMs and their related Resources corresponding to a CIMI `System` or `SystemTemplate` (see clause 5.13.1) and related CIMI management resources.

OVF support is covered in more detail in ANNEX A.

5 Model

This model assumes that a business relationship has already been established between the Consumer and the Provider. This relationship may include financial terms, creating separately administered clouds that the consuming organization is paying for, and the establishment of authentication credentials to access the administrative entry point for each cloud. The scope of this model is one separately administered cloud.

The CIMI model is described here by using a tabular representation. Each table is modeling a significant cloud resource for which independent access and manipulation is expected. Relationships between resources use a referential mechanism based on unique identifiers that is expected to be already supported by the implementation environment and protocol (e.g., URIs for HTTP).

The model is self-describing and allows for querying its own metadata, e.g., to discover which extensions have been implemented. The model is also extensible in different ways (see clause 5.1).

5.1 Extensibility

There are two types of extensibility mechanisms defined by the CIMI model; one is intended for use by Consumers whilst the other is to be used by Providers.

The first allows for a CIMI Consumer to add additional data to a Resource. Each Resource in the CIMI model has an attribute called "`properties`". Consumers, when creating or updating a Resource, may store any name/value pair in the `properties` attribute. CIMI Providers shall store and return these values to the Consumer. There is no obligation for the Provider to understand or take any action based on these values; they are there for the Consumer's convenience. Providers shall not add elements to this `properties` attribute.

The second type of extensibility mechanism allows for Provider defined extensions and this specification includes the `ResourceMetadata` Resource for this purpose. `ResourceMetadata` may be used to

- express constraints on the existing CIMI defined Resource attributes (e.g., express a maximum for the 'cpu' attribute of the `MachineConfiguration` Resource)
- introduce new attributes for CIMI defined Resources together with any constraints governing these (e.g., a new 'location' attribute for the `Volume` Resource that takes values from a defined set of strings)
- introduce new operations for any of the CIMI defined Resources (e.g., define a new 'compress' operation for the `Volume` Resource)
- express any Provider specific capabilities or features (e.g., the length of time that a `Job` Resource is retained after `Job` completion and before this is deleted)

It is recommended that Providers use the `ResourceMetadata` Resource to advertise these attributes, operations, and capabilities along with any constraints that might need to be understood by Consumers. The `ResourceMetadata` Resource is defined in clause 5.8.

5.2 Identifiers

All identifiers (e.g., Resource names, attributes, operations, parameter names) defined by this specification, or defined by the way of an extension, shall adhere to the following rules:

- Identifier names shall be treated as case sensitive.

- Identifier names shall only use the following set of characters:
  - Uppercase ASCII (U+0041 through U+005A)
  - Lowercase ASCII (U+061 through U+007A)
  - Digits (U+0030 through U+0039)
  - Underscore (U+005F)
- Identifier names shall not start with a Digit (U+0030 through U+0039).

Note that these rules do not apply to the "name" common attribute defined in clause 5.7.1.

5.3 Attribute constraints

Each attribute of any Resource is further qualified by a set of Boolean constraints. In particular, These constraints govern the level of support and access for an attribute, for either the Provider or the Consumer. Such constraints may be explicitly stated in the model itself for some Resources (i.e. determined by this specification), but in general are specified in the metadata Resource associated with a Resource (i.e. configured in the implementation).   These constraints are:

providerMandatory: (true/false)

If 'true', indicates that the attribute shall be supported by the Provider, i.e. always included as part of the Resource representation sent from Providers to Consumers, except if the attribute is empty. See clause 5.5.15 regarding empty attribute values. If present on a nested attribute, this attribute is required to be supported only if the parent attribute is supported. Default is 'true'.

consumerMandatory: (true/false)

If 'true', indicates, the attribute shall always be supported by the Consumer when using such a Resource, i.e. included as part of the Resource representation sent from Consumers to Providers, except if the attribute is empty. See clause 5.5.15 regarding empty attribute values. If present on a nested attribute, this attribute is required to be supported only if the parent attribute is supported. Default is 'false'.

mutable: (true/false)

If 'true', indicates that the attribute may be modified after initial creation of the Resource.If 'false', the attribute value will never change until the Resource is deleted. When the constrained attribute is a reference to another Resource, mutable = 'false' only means this reference will never change. It does not prevent updates on the referenced resource itself. Note that mutable = 'false' also implies consumerWritable = 'false'. Default is 'true'.

consumerWritable: (true/false)

If 'true' – and if mutable is also 'true' - indicates that the attribute may be directly set or updated by Consumers (update request), after creation of the Resource. Note that some Consumer operations on the Resource may have the indirect effect of changing some attribute values (this is obvious for the `updated` attribute, for example, or for the `state` of a Resource), but these are not considered as "direct" updates. Consequently such indirect updates are not precluded by consumerWritable = 'false'. Also, when the constrained attribute is a reference to another Resource, consumerWritable = 'false' only means this reference cannot be changed by the Consumer. It does not prevent updates on the referenced resource itself. Default is 'true'.

Additional requirements for Provider and Consumer:

- If a Provider receives a message containing an unknown or unsupported attribute, it shall reject the request.
- If a Consumer receives a message containing an unknown or unsupported attribute, it shall silently ignore the attribute. However, Consumers are required to include those attributes in

messages sent back to the Provider. Note in these cases the Consumer is not required to understand or process the unsupported attribute, but merely echo it back to the Provider

5.4 Serialization of Resources

The serialization of Resource instances in the model follow these conventions. Consider the serialization of a Resource named “`MyResource`”:

JSON serialization:

The Resource is serialized as an object wrapping all its attributes, but without a wrapper name. The Resource includes a `resourceURI` with a URI for the type of Resource being serialized. For example:

```
{ "resourceURI": "http://example.com/MyResource",
  "attribute": "value"
}
```

XML serialization:

The Resource is serialized as an element with name equal to the Resource name; for example:

```
<MyResource xmlns="http://example.com">
  <attribute> value </attribute>
</MyResource>
```

The serialization of attributes in a Resource follows the rules for the serialization of each data type, listed in section 5.5.

5.5 Data types and their serialization

Unless specifically asked to not include certain attributes in the Resource representation, the absence of an optional attribute in the representation means that the attribute has no value (i.e., is undefined), meaning there is no notion of an optional attribute having an implied value. Note that a client cannot distinguish (from just looking at the returned representation) whether a particular attribute is not supported from one that does not exist. Likewise, an absent attribute from a Resource representation as the input to an update operation means that the Consumer is requesting that the Provider remove that attribute.

The following clauses describe the data types and values that are used within the model definition tables.

5.5.1 boolean

A value as defined by xs:boolean per XML Schema – Part 2, with the exception that the only allowable values are either "true" or "false." The value is case sensitive.

If serialized in JSON, these values shall be of JSON type: *boolean*

If serialized in XML, these values shall be of XML Schema type: *xs:boolean*

5.5.2 dateTime

A value as defined by xs:dateTime per XML Schema – Part 2, which is consistent with DMTF DSP4004 and ISO 8601. The timestamp should preserve time zone information, i.e., include a local time component and an offset from UTC.

Any constraints on the specific ranges allowed for any particular attribute are specified by that attribute's definition or at runtime by the Provider by the way of the metadata discovery mechanisms defined by this specification.

For example, Monday, May 25, 2012, at 1:30:15 PM EST is represented as:

```
2012-05-25T13:30:15-05:00
```

If serialized in JSON, these values shall be of JSON type: *string*

If serialized in XML, these values shall be of XML Schema type: *xs:dateTime*

5.5.3 duration

A value as defined by xs:duration per XML Schema – Part 2. Any constraints on the specific ranges allowed for any particular attribute shall be specified by that attribute's definition or at runtime by the Provider by the way of the metadata discovery mechanisms defined by this specification.

If serialized in JSON, these values shall be of JSON type: *string*

If serialized in XML, these values shall be of XML Schema type: *xs:duration*

5.5.4 integer

A value as defined by xs:integer per XML Schema – Part 2. Any constraints on the specific ranges allowed for any particular attribute shall be specified by that attribute's definition or at runtime by the Provider by the way of the metadata discovery mechanisms defined by this specification.

If serialized in JSON, these values shall be of JSON type: *number*

If serialized in XML, these values shall be of XML Schema type: *xs:integer*

5.5.5 string

A value as defined by xs:string per XML Schema – Part 2. Any constraints on this type for any particular attribute shall be specified by that attribute's definition or at runtime by the Provider by the way of the metadata discovery mechanisms defined by this specification.

If serialized in JSON, these values shall be of JSON type: *string*

If serialized in XML, these values shall be of XML Schema type: *xs:string*

If serializing an attribute of type string, the serialization shall omit this attribute in case of an empty string.

5.5.6 ref

A reference to another Resource.

References allow for Consumers to navigate to Resources. By starting at the Cloud Entry Point and following the references that appear in the retrieved Resources, Consumers are able to recursively discover and navigate to all other Resources.

As a general rule, if an attribute is of type "`ref`", its value shall be held by an attribute named "`href`" (both in JSON and XML).

JSON serialization:

In the JSON serialization the `href` property appears as of type "`string`." If an attribute is of type "`ref`", the name of this attribute shall appear as a key, with the `href` property as a nested value. For example, a Resource attribute "`myvolume`" of type "`ref`" is serialized as:

```
"myvolume": { "href": string }
```

XML serialization:

In the XML serialization the `href` attribute appears as type "`xs:anyURI`." If an attribute is of type "`ref`," the name of this attribute shall appear as name of an XML element with the `href` property as an (XML) attribute. For example, a Resource attribute "`myvolume`" of type "`ref`" is serialized as:

```
<myvolume href="xs:anyURI"/>
```

References in both JSON and XML have an extensibility point that allows for additional information (such as the target Resource to be included "by value") if supported. For convenience, the JSON and XML representations, as shown above, exclude the implicit extensibility points that would allow for the attributes of the target Resource to be included if desired. So, more accurately the above representations might be written as follows:

For JSON:

```
"myvolume": { "href": string, ... }
```

and in XML:

```
<myvolume href="xs:anyURI"> xs:any* </myvolume>
```

However, for brevity the extensibility points are excluded from the serialization of the Resources.

5.5.7 map

A list of key/value pairs. The same "key" shall not be used more than once within an attribute. The "key" is case sensitive.

If serializing an attribute of type map, the serialization shall omit this attribute in case of an empty map.

5.5.8 structure

Attributes of this type are complex attributes made up of a set of nested attributes. For each attribute of this type, there is an additional table defining those nested attributes.

A nested structure can be considered a complex type definition. Structures may be named or unnamed. **Error! Reference source not found.** is an example of named structure:

Table 2 – Named structure

Name	*summary*	
Attribute	**Type**	**Description**
low	*number*	Number of “low” occurrences
medium	*number*	Number of “medium” occurrences
high	*number*	Number of “high” occurrences
critical	*number*	Number of “critical” occurrences

JSON serialization:

In JSON, the name of the structure (i.e., of the type it represents) never appears. In other words, whether the structure is named or not does not matter. An attribute named “`systemIncidents`” of type "`summary`" (as above) is serialized as follows:

```
"systemIncidents": {
  "low": number,
  "medium": number,
  "high": number,
    "critical": number
}
```

XML serialization:

In XML, the name of the structure (i.e., of the type it represents) never appears. In other words, whether the structure is named or not does not matter. The same previous "`systemIncidents`" example is serialized so that the structure sub-attributes become XML attributes of a <`systemIncidents`> XML element wrapper:

```
<systemIncidents low="xs:integer" medium="xs:integer" high="xs:integer"
        critical="xs:integer"/>
```

NOTE A large number of sub-attributes of atomic type in a structure may be represented alternatively as XML child elements for better readability. Both options are available; however, the same structure shall be serialized the same way across Resources.

5.5.9 byte[ ]

An arbitrary set of bytes meant to represent a block of binary data. Any constraints on this type for any particular attribute shall be specified by that attribute's definition or at runtime by the Provider by the way of the metadata discovery mechanisms defined by this specification.

If serialized in JSON, these values shall be of JSON type: *string*

If serialized in XML, these values shall be of XML Schema type: *xs:hexBinary*

5.5.10 URI

The format and syntax of the attributes of type "URI" is defined by RFC3986.

Unless otherwise noted, this specification does not mandate whether Providers use relative or absolute URI in the HTTP response bodies.

If URIs are specified as relative URIs, they shall be relative to the `baseURI`.

The algorithm used for converting a relative URI to an absolute URI shall be as described in section 5.2 of RFC3986. **Error! Reference source not found.** illustrates how relative URIs are resolved against base URIs:

Table 3 – Converting a relative URI to an absolute URI

Base URI	Relative URI	Absolute URI
`http://example.com/`	`p1/file`	`http://example.com/p1/file`
`http://example.com/c1/`	`p1/file`	`http://example.com/c1/p1/file`
`http://example.com/c1/c2/`	`p1/file`	`http://example.com/c1/c2/p1/file`

If relative URIs are used, the `baseURI` shall end with a trailing slash and relative URIs shall not begin with a leading slash. This format is consistent with most URI resolve utilities and produces the same results as a simple string concatenation algorithm.

If serialized in JSON, these values shall be of JSON type: *string*

If serialized in XML, these values shall be of XML Schema type: *xs:anyURI*

5.5.11 Array

An array represents an ordered list of items of the same type. An array shall always appear as an attribute of a Resource, and is only accessible as such (it is not a separately addressable Resource). If a

Resource is deleted, the items in its arrays shall also be deleted. However, in case these items were just references to other Resources, these referred Resources are not affected. (See the semantics of references in 5.7.)

Attributes that are arrays are defined by using the notation `itemType[]`, where `itemType` is the type name for each item of the array. If the type is a structure, not a simple data type, it is recommended as a convention in the model that the name of an array be the plural of a name that characterizes each item. For example, an array of volume items or of references to these may be named "volumes."

JSON serialization:

Within this specification, arrays in JSON are serialized with a wrapper property. The wrapper name shall be same as the attribute name for the array. For example, a "`things`" attribute of type "`thing[]`" is serialized as:

```
"things" : [
  { ... }, +
] ?
```

If the items in the array are structures, the structure name shall not be present in the JSON serialization.

In the case of an array of references, i.e., where the "`ref`" type applies to each element of the array, each element shall simply be serialized as an `href` property within a JSON array. For example, an array "`things`" of type "`ref[]`" is serialized as:

```
"things": [
  { "href": string },  +
] ?
```

NOTE If serializing arrays, conformant implementations shall not include empty arrays (i.e., arrays that contain no child properties) in the JSON serialization. Notice that the child of the "`things`" property is defined with a "+", meaning at least one child is required. This requirement ensures that the JSON serialization is minimized and only includes the wrapping "`things`" element if, and only if, there is at least one "`thing`" in the array.

XML serialization:

The XML serialization of arrays requires each item of the array to be represented as an element. These elements shall be consecutive and contiguous in the serialization and the name of each element (tag name) shall be the name of the element type (the name that appears before "`[]`" in the array type). As in JSON, the serialized array has a wrapper element of same name as the array attribute name. For example, a "`things`" attribute shall be serialized as a list of items named "`thing`":

```
<things>
   <thing>
  ...
   </thing> *
</things>

```

In the case of an array of references, i.e., where the "`ref`" type applies to each element of the array, the array is serialized as a list of XML elements without wrapper. Each element is named per an array “item name" specified in the attribute's definition. For example, an array "`things`" of type "`ref[]`" where the array “item name" is "`thing`" is serialized as:

```
<thing href="xs:anyURI"/> +
```

5.5.12 Collection

A Collection is a group of Resources of the same type. In contrast with arrays, Collections are themselves Resources that have their own URI and can be independently accessed. Collections also allow for an optimized and convenient interaction pattern by providing a specialized set of operations that avoid replacing a large number of items when updating the set, as with arrays.

This specification uses Collections if the set of grouped items is modified often and potentially by multiple Consumers. Conversely, arrays are used if it is expected that the list of items is not modified often or can be easily modified by substitution of the entire list, and thus the overhead of managing these items as separate Resources might be unjustified and burdensome.

An item in a Collection, i.e. a Collection item, is an embedded structure that contains a reference to a Resource and optionally additional attributes (see “accessory” attributes, defined later). For convenience, the Resource referred to by a Collection item is called here a Resource item of the Collection.

A Resource may be referenced by more than one Collection. If such a Resource is deleted, every Collection that references this Resource shall remove the corresponding item. While different Collections contain entries of different Resource types, all Collections follow the pattern described below:

- A Collection shall contain an `id` attribute that acts as a "self pointer." Retrieving the data at this reference shall return the Collection. In the XML representation, each Collection shall be wrapped by a `<Collection>` element.
- A Collection shall contain a `count` attribute that indicates the number of Resources in the Collection at the time the Collection was queried.
- Adding new Resources to the Collection shall be done either via the "add" operation defined within the Collection (when the Resource is also created) or via the “insert” operation (when the Resource already exists).

Deleting an item from the Collection shall be done either via a "delete" operation on the Resource item itself (if the Resource has to be discarded) or via the “remove” Collection operation (if the Resource must still exist outside the Collection).Collections that are attributes of other Resources are represented with attribute type "`collection[itemType]`." The Resource type of the Collection items are specified inside the brackets; for example an attribute that is a Collection of Machines is expressed as "`collection[Machine]`." Attributes of such types are serialized as a reference to a Collection Resource instead of holding the Collection itself as value. For brevity, while these attributes are "references" the word "ref" or "reference" does not appear in the model definition tables - instead the type of such an attribute is making abstraction of the reference and more explicitly shows as "`collection[itemType]`".

In the serializations below, the Collection items are represented by items in the *ResourceSpecificGroupingName* JSON array, and by *ResourceSpecificElementName* elements *in the* XML representation*.*

Serialization:

The serialization of Collections shall adhere to the following pattern:

JSON serialization:

```
{ "resourceURI": string,
  "id": string,
  “updated": string, ?
  "parent": string, ?
  "count": number,
  "resourceSpecificGroupingName": [
    { "resourceURI": string,
      "id": string,
      "name": string, ?
      "description": string, ?
      "created": string, ?
      "updated": string, ?
      "parent": string, ?
      "properties": { string: string, + }, ?
      ... resource specific data ...
      "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="xs:anyURI" xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <updated> xs:dateTime </updated> ?
  <parent> xs:anyURI </parent> ?
  <count> xs:integer </count>
  <resourceSpecificGroupingName>
    <ResourceSpecificElementName>
      <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> *
    ... resource specific data ...
```

```
        <operations>
          <operation rel="edit" href="xs:anyURI"/> ?
          <operation rel="delete" href="xs:anyURI"/> ?
        </operations>
        <xs:any>*
      </ResourceSpecificElementName> *
    </resourceSpecificGroupingName>

    <operations>
      <operation rel="add" href="xs:anyURI"/> ?
      <operation rel="insert" href="xs:anyURI"/> ?
      <operation rel="remove" href="xs:anyURI"/> ?
    </operations>
    <xs:any>*
  </Collection>
```

Where the `resourceURI` attributes shall contain the Collection or Resource specific URIs for that type of Collection, and `resourceSpecificGroupingName` and `ResourceSpecificElementName` shall be replaced with the name of the Collection-specific Resource name, e.g., `machines` in JSON or `Machine` in XML.

The above serialization shows that each entry in a Collection may contain "resource specific data" beside the reference to the Resource item and the common attributes. This placeholder represents two kinds of data:

a) Optionally some *accessory attributes* that represent accessory information for the use of this reference in the context of the Resource owning that Collection (the accessory attributes) – e.g., the "initial location" of a referenced `Volume`, in a Collection of Volumes associated with a `Machine`. Accessory attributes – if any - are part of the definition of each specific Collection..

b) All or a subset of the attributes of the corresponding Resource items. How much of the Resource item is expanded in the serialization of the Collection is controlled by expansion mechanisms described later.

If accessory attributes exist for items in a Collection, the "*resourceSpecificGroupingName" or "ResourceSpecificElementName"* is not just identifying the Resource type of Collection items, but is a unique name specific to this combination of accessory attributes and Resource type – e.g., for Volumes with initial location, it may be "locatedVolume". Also the resourceURI of the Collection is unique to this combination. Because of this accessory attribute, the Collection of Volumes is said to be "enhanced", as opposed to "basic" for a Collection without accessory attribute.

The serialization of Collections follows these additional rules:

- A Provider may limit the number of Resources returned in the Collection. The Consumer can determine this has occurred by comparing the number of returned Resources with the value of the "Count" attribute and any Collection subsetting query parameters it specified. In this case, the Consumer is advised to specify filter query parameters (see 4.1.6.1) to reduce the number of entries returned, or retrieve them in batches by issuing multiple requests with Collection subsetting query parameters (see 4.1.6.2)

- As with all Resources in the CIMI model, each Resource in the Collection shall have an `id` attribute that acts as a "self pointer." Retrieving the data at this reference shall return just that one Resource and not any parent Resource, such as the Collection or array attribute.
- The serialization of a Collection may be controlled (see 4.1.6.4 `$expand` query parameter) to show more or less of each Resource item. By default, each entry in the Collection will show just a reference (URL) to the Resource item, along with the “common” attributes of the Resource item. Alternatively, the Resource item may be expanded partially or fully when querying the Collection.
- As with all arrays, if there are no Resources in the Collection, the serialization of the list shall be omitted.

5.5.12.1 Adding an item to a Collection

Invoking the "add" operation of a Collection shall create a new Resource and add it to the Collection. The contents of the request body shall be either a representation of the new Resource being added to the Collection, or a representation of the Template associated with the new Resource being created and resource specific data attributes.

If creating a new Resource the "add" operation shall contain:

- The "common attributes" as defined by clause 5.7.1
- The Resource specific data needed to create it. This data shall either be a reference to the Resource-specific Template Resource or the Resource-specific Template Resource itself inlined.
- Accessory attributes–if any–that represent accessory information for the use of the reference in the context of the Resource owning that Collection (the associative attributes)
- In the XML case, a wrapper element (named after the pattern <*ResourceName*Create>)

For example, to create a new `Machine` (which requires the use of a Template) and add it to the `MachineCollection`, the "add" operation of the `MachineCollection` shall be serialized as follows:

JSON serialization:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/MachineCreate", ?
  "name": string, ?
  "description": string, ?
  "properties": { string: string, + }, ?
  "machineTemplate": { "href": string ?}
  ...
}
```

XML serialization:

```
<MachineCreate xmlns="http://schemas.dmtf.org/cimi/2">
  <name> xs:string </name> ?
  <description> xs:string </description> ?
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties>
  <machineTemplate href="xs:anyURI"? />
  <xs:any>*
</MachineCreate>
```

The `MachineCollection` has a new `Machine`:

JSON serialization:

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

XML serialization:

```
<Machine xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <name> xs:string </name>
  ...
</Machine>
```

The processing of the "add" operation shall adhere to the semantics defined in clause 4.2.1.1.

Regardless of whether a Template is used, the "add" operation shall create the new Resource and add it to the Collection and a reference (URI) to the new entry shall be returned in the response message in the HTTP Location header.

5.5.12.2 Inserting an item in a Collection

Invoking the "insert" operation of a Collection shall add to the Collection a new reference to an existing Resource. The contents of the request body shall specify the URL of the existing Resource being added and the accessory attributes in case of an “enhanced” collection.

In order to add an existing `Volume` to the `volumes` Collection of a `Machine`, the request body of the "insert" operation shall be serialized as follows:

JSON serialization:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/Action",
  "action": “http://schemas.dmtf.org/cimi/2/action/insert”,
  "initialLocation": string,
  "volume": { "href": string }
}
```

XML serialization:

```
<Action xmlns="http://schemas.dmtf.org/cimi/2">
  <action>http://schemas.dmtf.org/cimi/2/action/insert</action>
  <initialLocation> xs:string </initialLocation>
  <volume href="xs:string"/>
</Action>
```

Note that “`initialLocation`” is an accessory attribute to each reference of Volume. Because of this addition, the type of the collection items is distinguished from `Volume`, and called here `locatedVolume`. The definition of the `volumes` Collection of the `Machine` Resource describes the accessory attribute(s) for this Collection.

5.5.12.3 Removing an item from a Collection

Invoking the "remove" operation of a Collection shall delete the specified item in the Collection, i.e. the Resource reference along with accessory attributes if any, without destroying the referenced Resource item itself. The contents of the request body shall be the URL of the Resource item being removed.

In order to remove a `Volume` from the `volumes` Collection of a `Machine`, the request body of the "remove" operation shall be serialized as follows:

JSON serialization:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/Action",
  "action": “http://schemas.dmtf.org/cimi/2/action/remove”,
  "volume": { "href": string }
}
```

XML serialization:

```
<Action xmlns="http://schemas.dmtf.org/cimi/2">
  <action>http://schemas.dmtf.org/cimi/2/action/remove</action>
    <volume href="xs:string"/>
</Action>
```

Removing the referenced Resource (here a `Volume`) deletes the related entry from the Collection. This deletes the reference but not the Resource itself.

5.5.12.4 Deleting an item in a Collection

Deleting the Resource referenced by a Collection item via a DELETE operation on the Resource itself (in the previous example, a `Volume`) also deletes the related entry from the Collections that reference this Resource – i.e., it has the effect of a “remove” on the Collection, in addition to deleting the referenced Resource.

5.5.13 "Any" type

Some attributes are polymorphic and can hold various data types, the list of which is indicated in their description. In such cases, the type of the attribute shall be indicated as "any" in the model representation.

5.5.14 valueScope

The valueScope type is a specialized map. Its goal is to define possible values for a list of attributes of a Resource. The possible values for an attribute are called the “value scope” of the attribute, and a combination of attribute value scopes (in form of a map) in a Resource or in the ResourceMetadata is called the value scope of the Resource.

Each item in a valueScope is a key/value pair where:

- The key is the name of an attribute of a Resource – or “**scoped attribute**” – for which a set of possible values is defined.
- The value is a structure that defines the “**scope**”, i.e., a range, an enumeration or a single assigned value for the scoped attribute.

The scope structure:

A “scope” structure – or the value part of a key-value item in a valueScope – can take one of following forms:

1) An assigned single value, along with its (optional) `units`, e.g., for a scoped attribute named “cpu”:

```
"cpu": { "value": 2000, "units": “megahertz” }
```

In the above, `value` and `units` are reserved keywords for defining the value scope.

2) A range of values, along with its optional `units`, and an optional `increment` e.g., for a scoped attribute named "memory". The range may be open-ended: either the `minimum` or the `maximum` may be missing. The `increment` specifies the allowed values starting from the `minimum` and upward - i.e., the allowed values are of the form: minimum+N*(increment), where N>=0, or starting from the `maximum` and downward in case there is no `minimum,` i.e., allowed values are of the form: maximum-N*(increment),.

```
"memory": { "minimum": 4000, "maximum": 10000, "units": “kibibytes”, "default":
4000, "increment": 2000 }
```

In the above, `minimum, maximum, default, increment` and `units` are reserved keywords for defining the value scope.

3) An enumeration (or `values`), along with its (optional) `units`, e.g., for a scoped attribute named “cpuArch”:

```
"cpuArch": { "values": [ “68000”, “Alpha”, “ARM”, “ PA_RISC”], "default": “PA_RISC”
}
```

4) Simply a required `units`, e.g., for a scoped attribute named “capacity”:

```
"capacity": { "units": “megabytes” }
```

5) Any of the above, applying to the items in a collection, e.g., for a range of values that applies to the accessory attribute named “`remoteLocation`” of type URI for every item in a collection named `machines`:

```
"machines": { "item": {“remoteLocation”: { "values": [ “URI1”, “URI2”, “URI3”],
"default": “URI1” }}}
```

In the above, `item, values` and `default` are reserved keywords for defining the value scope.

If a valueScope is associated with a Resource type, it shall be in form of an attribute named “vscope”, of type array of valueScope (i.e., valueScope[]).

An example of valueScope for the MachineConfiguration Resource:

```
"vscope" : [ {

"cpu": { "value": 1 },

"memory": { "minimum": 4, "maximum": 32, "units": “GbB”, "default": 4, “increment”:
2 },

"cpuArch": { "values": [ “68000”, “Alpha”, “ARM”, “ PA_RISC”, “i5”], "default":
“i5” }

} ]
```

Semantics of valueScope array in a Resource

The value scope of a Resource shall be represented by an array of valueScope instances, even if in many cases this array will contain a single valueScope instance. This allows for expressing dependencies between values of different attributes of a same Resource. In such cases, the scoped attributes of the Resource must satisfy either valueScope instance in this array.

In the following example, `vscope` is an array of two valueScope items:

```
"vscope":  [ {
"cpuSpeed": { "minimum": 2, "maximum": 4, "units": "GHz", "default": 2.5},
"memory": {"minimum": 2000000, "maximum": 10000000, "units": "KbB", "increment":
2000000 },
"cpuArch": { "value": "i5" }
}, {
"memory": { "minimum": 4000000, "maximum": 32000000, "units": "KbB" },
"cpuArch": { "values": [ "68000", "Alpha", " PA_RISC"] }
} ]
```

This valueScope means that the Provider supports MachineConfigurations with either `cpuArch` of value `i5`, or of a value that is one of { `"68000", "Alpha", " PA_RISC" }.` In the first case (`i5`), the memory must be within the 2GbB-10GbB range and `cpuSpeed` must be between 2-4 GHz, while in the second case the memory must be within the 4GbB-32GbB range.

The following pseudo-schemas describe the serialization of the valueScope map in both JSON and XML:

JSON serialization:

```
( "value": any,
"units": string ? ) |
( "values": [ any,+ ],
"units": string ,?
"default": string ? ) |
( "minimum": number, ?
"maximum": number, ?
"units": string ,?
"default": number, ?
"increment": number ? )

```

XML serialization:

```
( <value> xs:any </value>
<units> xs:string </units> ? ) |
(<value> xs:any </value> +
<units> xs:string </units> ?
<default> xs:any </default> ? ) |
(<minimum> xs:integer </minimum> ?
<maximum> xs:integer </maximum> ?
```

```
        <units> xs:string </units> ?
        <default> xs:integer </default> ?
        <increment> xs:integer </increment> ? )
```

A Provider who supports value scopes shall set the ValueScopes capability (ResourceMetadata) to “true”.

5.5.15 Empty attribute values

Attributes of the following types are omitted in cases where they have an empty value: string, map, array, and Collection. Apart from being “Provider optional” or “Consumer optional”, an empty value is the third reason that the serialization schema contains an ‘?’ or an ‘*’ for an attribute.

Other attribute types do not have empty values and shall not be omitted from the serialization for this reason.

5.6 Units

Some of the Resources defined by this specification have attributes that describe an amount of something that belongs to, or is associated with, that Resource. For example, the `Machine` Resource has a `memory` attribute that describes "the size of the memory allocated to this machine." The allowable units of these attributes are listed in **Error! Reference source not found.**. Their meaning is defined in IEC 80000-13:2008. Their numerical equivalents are provided here for convenience:

Table 4 – Numerical equivalents for attributes

String	Numerical Value	String	Numerical Value
kilobyte	10^3	kibibyte	2^10
megabyte	10^6	mebibyte	2^20
gigabyte	10^9	gibibyte	2^30
terabyte	10^12	tebibyte	2^40
petabyte	10^15	pebibyte	2^50
exabyte	10^18	exbibyte	2^60
zettabyte	10^21	zebibyte	2^70
yottabye	10^24	yobibyte	2^80

5.7 Resources

CIMI Resources are representations of actual – either virtual or physical – resources available in a Cloud. Resources are identified and separately accessible by their URI. Every Resource has a type which is described in this section. A Resource type defines a set of attributes and of operations.

5.7.1 Common Resource attributes

Resources, except for the Collection Resource, shall support the following common attributes defined in **Error! Reference source not found.**.A Collection Resource shall support the `id` attribute, the `updated` attribute and the `parent` attribute, as defined in Table 5.

Table 5 – Common attributes

Attribute	Type	Description
id	*URI*	The unique URI identifying this Resource; assigned upon Resource creation. This attribute value shall be **unique** in the Provider’s cloud. **<u>Constraints:</u>** providerMandatory: true consumerMandatory: true mutable: false consumerWritable: false
name	*string*	The human-readable name of this Resource; assigned by the creator

Attribute	Type	Description
		as a part of the Resource creation input. **<u>Constraints:</u>** providerMandatory: true consumerMandatory: false mutable: true consumerWritable: true
description	*string*	The human-readable description of this Resource; assigned by the creator as a part of the Resource creation input. **<u>Constraints:</u>** providerMandatory: true consumerMandatory: false mutable: true consumerWritable: true
created	*dateTime*	The timestamp when this Resource was created. The format should be unambiguous, and the value is **immutable**. **<u>Constraints:</u>** providerMandatory: false consumerMandatory: false mutable: false consumerWritable: false
updated	*dateTime*	The time at which the last explicit attribute update was made on the Resource. The initial value is the time the resource is created. Note, while operations, such as "stop", do implicitly modify the 'state' attribute, they do not change the 'updated' time. **<u>Constraints:</u>** providerMandatory: false consumerMandatory: false mutable: true consumerWritable: false
parent	*ref*	A reference to a Resource of which this Resource is a child component (see “composition” relationship, section 5.10.2) – i.e. a reference to its first parent Resource. **<u>Constraints:</u>** providerMandatory: true consumerMandatory: false mutable: true consumerWritable: true
properties	*map*	A map of key/value pairs (each entry called a "property"), some of which may control one or more aspects this Resource. Properties may also serve as an extension point, allowing Consumers to record additional information about the Resource. The same "key" shall not be used more than once within a "properties" attribute. Each property shall contain the following nested data: **Name**: *property* **Data** | **Type** | **Description** key | *string* | The name of the property. value | *string* | The value of the property. **<u>Constraints:</u>** providerMandatory: false consumerMandatory: false mutable: true consumerWritable: true

Attribute	Type	Description
resourceMetadata	*ref*	A reference to a ResourceMetadata instance associated with this Resource and governing the attributes, operations and capabilities concerning this Resource. **Constraints:** providerMandatory: false consumerMandatory: false mutable: true consumerWritable: false

The following pseudo-schemas describe the serialization of these attributes in both JSON and XML:

JSON serialization:

```
"id": string,
"name": string, ?
"description": string, ?
"created": string, ?
"updated": string, ?
"properties": { string: string, + }, ?
"resourceMetadata" : ["href": string, * ], ?
```

XML serialization:

```
<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> ?
<resourceMetadata href=”xs:string” />  ?
```

5.8 Operations

All Resource operations defined by this specification are optional for Providers to support. Consumers, by the way of examination of a Resource's ResourceMetadata, can determine which operations are supported. However, even for those operations that are supported Consumers still need to examine each Resource's representation to determine which operations are supported at that moment. Whether an operation is supported is based on a number of factors, including the state of the Resource and access control rights of the Consumer. Also see clause 4.2. Operations and states are coupled; i.e., if implementing a state-changing Resource operation defined in this specification, the corresponding state(s) shall also be implemented. See the Resource-specific “Operations” clauses for additional detail.

The "State" attribute of Resources that have this attribute shall only change value if

- an operation is performed on this Resource and this operation requires a state change, or
- an error occurred, in this case the “State” attribute shall obtain the value “ERROR”.

For example, for a 'start' operation on a `Machine` both the STARTING and the STARTED states are required to be supported by the `Machine`, while the `Machine` can only leave the STARTED state after another state changing operation is requested, unless an error occurs.

Providers can define additional operations and states. Such extensions shall fall into one of these categories:

a) A new operation that starts from a CIMI-defined state, or leads to a CIMI-defined state, or both. In the latter case, if a CIMI-defined operation already exists for this transition between two CIMI-defined states, it shall also be supported by the Provider in addition to the new operation.

b) A new Resource state. In that case, a new operation that leads to that state shall also be created. In other words, a Provider-defined operation has to be performed before a Provider-defined state can be reached.

c) A new operation that transitions between two Provider-defined states.

5.9 Alternative model formats

It is expected that this specification is implemented by using a variety of technologies. As a convenience, the definition of the model elements are provided in alternative formats that are easily consumable by technology-specific tooling.

In the event of inconsistencies between the various formats, the normative text within this specification takes precedence over the XML Schemas and alternative formats, which in turn take precedence over examples.

5.10 Relationships between Resources

5.10.1 Referencing across Resources

Resources may refer each other. This referencing expresses a directional relationship in which there is a *referring* Resource and a *referred* Resource. Depending on the cardinality of such relationships, there are two representations:

- For 1-to-1 referencing, the URL of the referred Resource appears as an attribute in the referring Resource.
- For 1-to-n referencing, the referred Resources (all of the same type) are grouped in a Collection, the URL of which appears as an attribute in the referring Resource. In that case, the *referring* Resource does not refer directly to the referred Resources, but instead to a Collection Resource that contains references to the *referred* Resources.

If a *referred* Resource is deleted but not the *referring* Resource(s), then in case of a 1-to-1 relationship the reference shall be set to empty in every *referring* Resource, and in case of a 1-to-n relationship the reference shall be removed from any Collection where it appears as an item.

5.10.2 Composition Relationship between Resources

A Resource is a child component of another Resource if its `parent` attribute refers to the latter Resource. This relationship is transitive.

If a Resource is deleted, its child component Resource(s) is(are) also automatically deleted.

In case of a Collection Resource that is referred by a Resource R,.expressing a composition relationship from the Collection Resource items to R is done by:

(a) setting the `parent` attribute of each Resource item to the Collection Resource and

(b) by setting the `parent` attribute of the Collection Resource to the Resource R.

A Resource is said to be parent of its children components.

In any Resource description R throughout this specification, an attribute of type "collection[]" refers to a Collection Resource that has the Resource R as a parent, unless indicated otherwise.

For example a Machine is parent of its related Disk Resources via the `disks` Collection: the `parent` attribute of a Disk is set to the `disks` Collection, and the `parent` attribute of the `disks` Collection is set to the Machine.

Some composed Resources – e.g. System - may have component Resources that are not their "children". Such Resources are called associated components. Their `parent` attribute refers to another Resource or to the CEP, meaning the deletion of the composed Resource does not cause the deletion of its associated components, even if the associated components are still otherwise managed by the composed Resource.

5.11 Resource Metadata

Implementations of this specification should allow for Consumers to discover the metadata associated with any Resource under the Cloud Entry Point. Doing so allows for the discovery of Provider‑defined constraints on the attributes or operations of a Resource as well as discovery of any new extension attributes or operations that the Provider may have defined.

A `ResourceMetadata` instance contains metadata governing the attribute status (optionality, value constraints, access), the available operations, and other Provider-specific capabilities or features for a Resource or a set of Resources, called the target Resource(s) for that `ResourceMetadata` instance.The target Resource contains a reference to its `ResourceMetadata` instance, which itself may be shared across several target Resources.

Any Resource under a CEP may have a `ResourceMetadata` instance associated with it. This association may be done in one of the following ways:

1. A `ResourceMetadata` instance is defined for all Resources of a same type under the CEP. In such a case the `ResourceMetadata` instance is added as a Resource item in the resourceMetadata collection unique to the CEP. Unless overridden, it applies to all Resources of the targeted type under this CEP.
2. A `ResourceMetadata` instance is defined for all Resources generated from a same template. In such a case, a Template-specific `ResourceMetadata` instance is provided and referred by this Template. This `ResourceMetadata` overrides any CEP-level `ResourceMetadata` (1) for the type of Resource generated from this Template.
3. A `ResourceMetadata` instance may be created for a single particular Resource instance, or may be associated on a per-Resource basis. Such an association requires an explicit modification of the `resourceMetadata` attribute of the target Resource, canceling any former value it may have been given at creation time, e.g. in above cases (1) or (2)

Each Resource's metadata shall contain the following pieces of information:

Table 6 – ResourceMetadata attributes

Name	ResourceMetadata	
Type URI	http://schemas.dmtf.org/cimi/2/ResourceMetadata	
Attribute	**Type**	**Description**
typeURI	*URI*	A unique URI associated with, and denoting, the type of the described Resource target. **<u>Constraints:</u>** providerMandatory: true consumerMandatory: true mutable: true

Name	ResourceMetadata	
Type URI	http://schemas.dmtf.org/cimi/2/ResourceMetadata	
Attribute	**Type**	**Description**
		consumerWritable: true
name	*string*	The name of the Resource target type (e.g. Machine). **<u>Constraints:</u>** providerMandatory: true consumerMandatory: true mutable: true consumerWritable: true
attributes	*attribute[]*	A set of metadata associated with each attribute (or target attribute) of the Resource target, including the set of extension attributes not defined in this specification. The metadata for each attribute target shall contain the following nested data: (see nested table below) <u>Every above attribute in the nested `attribute` table has the following constraints:</u> providerMandatory: true consumerMandatory: true mutable: true consumerWritable: true The constraints for the `attributes` attribute of `ResourceMetadata` are: **<u>Constraints:</u>** providerMandatory: false consumerMandatory: false mutable: true consumerWritable: true
vscope	*valueScope[]*	The `vscope` attribute applies to the attributes of the described – or target – Resource. The target Resource shall be of the type identified by the `typeURI`

Name	*attribute*	
Data	**Type**	**Description**
name	*string*	The name of the target attribute.
namespace	*URI*	The namespace in which the target attribute is defined. It is recommended that a dereference of this URI returns information about the attribute. This shall not be present if describing a CIMI-defined attribute, but shall be present if describing a non-CIMI defined attribute (i.e. an extension).
type	*string*	The data type of the target attribute. This shall not be present if describing a CIMI-defined attribute, but shall be present if describing a non-CIMI-defined attribute (i.e. an extension).
provider Mandatory	*boolean*	If "true" (by default) Indicates that the target attribute shall be present in any representation of this Resource sent by a Provider (if it has a non-empty value). See more precise definition in 5.3.
consumer Mandatory	*boolean*	If "true" Indicates that the target attribute shall be present in any representation of this Resource sent by a Consumer. (if it has a non-empty value). Default is "false". See more precise definition in 5.3.
mutable	*boolean*	If "true" (by default) Indicates that the target attribute may be modified after the Resource creation. See more precise definition in 5.3.
consumer Writable	*boolean*	If "true" (by default) Indicates that the target attribute may be modified by the Consumer. See more precise definition in 5.3.

Name	ResourceMetadata	
Type URI	http://schemas.dmtf.org/cimi/2/ResourceMetadata	
Attribute	**Type**	**Description**
		attribute. Consequently this value scope is about the list of attributes described in the `attributes` attribute. If an attribute of the target Resource is constrained by the vscope, a Consumer shall set a value (creation or update request) compatible with the value scope of this attribute. For any other case where the Consumer sets an incompatible value, the Provider shall return a 4xx error code. **Constraints:** providerMandatory: false consumerMandatory: false mutable: true consumerWritable: true
capabilities	*capability[]*	A set of Provider-defined metadata that can be used by Consumer to discover any capability or feature provided by this Provider. Each capability shall contain the following nested data:

Name	*capability*	
Data	**Type**	**Description**
name	*string*	The name of the capability.
uri	*URI*	A URI that uniquely identifies the capability at a global level. **Constraints:** consumerMandatory: true
description	*string*	The human-readable description of the semantic of the capability.
value	*any*	The value of the capability. The specific type varies depending on the definition of the capability. If not present the capability defaults to a "boolean" type with a value of "true" indicating that the specific capability is supported by the Provider. **Constraints:** consumerMandatory: true

		Every above attribute in the nested `capability` table has the following constraints by default (unless overridden per attribute): providerMandatory: true consumerMandatory: false mutable: true consumerWritable: true The constraints for the `capabilities` attribute of `ResourceMetadata` are: **Constraints:** providerMandatory: false consumerMandatory: false mutable: true consumerWritable: true
actions	*action[]*	A set of Provider-defined operations that can be used by consumers to act on the Resource. This set represents all operations defined for this described Resource type, which may be a superset of those operations a particular Consumer is actually allowed to use. The subset of allowed operations for a particular Consumer shall be those operations returned to this Consumer if querying an instance of the described Resource type. Note that this attribute is called "actions" so as not to conflict with the ResourceMetadata Resource's own operations. Each operation shall contain the following nested data:

Name	*action*

Name	ResourceMetadata	
Type URI	http://schemas.dmtf.org/cimi/2/ResourceMetadata	
Attribute	**Type**	**Description**

Data	**Type**	**Description**
name	*string*	The name of the operation.
uri	*URI*	A URI that uniquely identifies the operation at a global level.
description	*string*	The human-readable description of the semantic of the operation. **Constraints:** consumerMandatory: false
method	*string*	The protocol-dependent verb to use to perform the operation.
inputMessage	*string*	The body mimeType of the request message; it may depend on the model format chosen by the Provider.
outputMessage	*string*	The body mimeType of the response message; it may depend on the model format chosen by the Provider.

Every above attribute in the nested `action` table has the following constraints by default (unless overridden per attribute):
providerMandatory: true
consumerMandatory: true
mutable: true
consumerWritable: true

The constraints for the `actions` attribute of `ResourceMetadata` are:
Constraints:
providerMandatory: false
consumerMandatory: false
mutable: true
consumerWritable: true

When implementing or using `ResourceMetadata`, Providers and Consumers shall adhere to the syntax and semantics of its attributes as described in **Error! Reference source not found.** as well as in the tables describing embedded Resources or related Collections. 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:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/ResourceMetadata",
  "id": string,
  "typeURI": string,
  "name": string,
  "attributes" : [
    { "name": string,
      "namespace": string, ?
      "type": string, ?
      "required": boolean, ? } *
        ], ?
  "vscope" : [ valueScope, * ], ?
  "capabilities": [
    { "name": string, ?
      "uri": string,
      "description": string, ?
      "value": any } *
  ], ?
  "actions" : [
    { "name": string,
      "uri": string,
      "description": string, ?
      "method": string,
      "inputMessage": string, ?
      "outputMessage": string ? }, *
  ], ?
  "operations": [
    { "rel": "edit", "href": string }, ?
    { "rel": "delete", "href": string } ?
  ] ?
  ...
}
```

XML media type: application/xml

XML serialization:

```
<ResourceMetadata xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <name> xs:string </name>
  <typeURI> xs:anyURI </typeURI>
  <attributes>
    <attribute name="xs:string" namespace="xs:anyURI"? type="xs:string"?
             required="xs:boolean"? /> *
      </attribute> *
  </attributes>
  <vscope> valueScope </vscope>?
  <capabilities>
    <capability name="xs:string"? uri="xs:anyURI" description="xs:string"?>
    xs:any*
    </capability> *
  </capabilities>
  <actions>
    <action name="xs:string" uri="xs:anyURI" description="xs:string"?
          method="xs:string" inputMessage="xs:string"?
          outputMessage="xs:string"? /> *
  </actions>
  <operations>
    <operation rel="edit" href="xs:anyURI"/> ?
    <operation rel="delete" href="xs:anyURI"/> ?
  </operations>
  <xs:any>*
</ResourceMetadata>
```

Additional metadata about the Resource or attributes may be included by the Provider.

5.11.1 Capabilities

Error! Reference source not found. describes the capability URIs defined by this specification. Providers may define new URIs and it is recommended that these URIs be dereferencable such that Consumers can discover the details of the new capability. The "Resource Name" column contains the name of the Resource that may contain the specified capability within its `ResourceMetadata`. The "Capability Name" column contains the name of the specified capability and shall be unique within the scope of the corresponding Resource. Each capability's URI shall be constructed by appending the "Resource Name", a slash (/), and the "Capability Name" to "http://schemas.dmtf.org/cimi/2/capability/". For example, the `Machine's` "InitialState" capability shall have a URI of:

```
http://schemas.dmtf.org/cimi/2/capability/Machine/InitialState
```

Capabilities that apply to the Provider in general, and are not specific to any one Resource, shall be associated with the `CloudEntryPoint` Resource (in case a capability applies only to the `CloudEntryPoint` Resource itself, its definition indicates this).

Each one of these capabilities may be set to some value, or may be absent. The meaning of an absent capability is defined as follows:

- For boolean-valued capabilities: same as a “false” value.
- For other capabilities that use a single value or a list of values among an enumeration: same as no particular preference or restriction being enforced for this value.

Table 7 – Capability URIs

Resource Name	Capability Name	Description
CloudEntryPoint	ExpandParameter	If true, the Provider shall support the `$expand` query parameter.
CloudEntryPoint	FilterParameter	If true, the Provider shall support the `$filter` query parameter.
CloudEntryPoint	FirstParameter	If true, the Provider shall support both the `$first` and `$last` query parameters.
CloudEntryPoint	SelectParameter	If true, the Provider shall support the `$select` query parameter.
CloudEntryPoint	FormatParameter	If true, the Provider shall support the `$format` query parameter.
CloudEntryPoint	OrderByParameter	If true, the Provider shall support the `$orderby` query parameter.

Resource Name	Capability Name	Description
CloudEntryPoint	QueryPathNotation	If true, the Provider shall support the use of path-like notation with query parameter `$select` (see 4.1.6.3) to disambiguate between attributes of a Collection Resource and attributes of each items in the Collection if subsetting.
CloudEntryPoint	MaxPropertyItems	If set, the Provider shall support a ‘Properties’ attribute with a number of elements less than or equal to the size specified by this capability.
CloudEntryPoint	ValueScopes	If true, the Provider shall support the use of attributes of type valueScope, for any Resource that may be created via a template.
System	SystemComponentTemplateByValue	If true, the Provider shall support the specification of ComponentTemplates by value in SystemTemplates.
Machine	DefaultInitialState	If this capability is set, unless otherwise provided (e.g., by a MachineTemplate "initialState" attribute), the Provider shall set a new Machine to this state value, assuming the value is compatible with the InitialStates capability, if set.
Machine	InitialStates	If this capability is set, and if using a MachineTemplate that has an “initialState” attribute, a Consumer shall use an initialState value from the set of values of this capability.
Machine	MachineConfigByValue	If true, the Provider shall support specifying MachineConfigurations by value. If true, the MachineTemplateByValue shall also have the value true.
Machine	MachineCredentialByValue	If true, the Provider shall support specifying Credentials by value in Machine create operations. If true, the MachineTemplateByValue capability shall also have the value true.
Machine	MachineImageByValue	If true, the Provider shall support specifying MachineImages by value in Machine create operations. If true, the MachineTemplateByValue capability shall also have the value true.
Machine	MachineVolumeTemplatesByValue	If true, the Provider shall support specifying VolumeTemplates by value in Machine create operations. If, then the MachineTemplateByValue capability shall also have the value true.
Machine	MachineTemplateByValue	If true, the Provider shall support specifying MachineTemplates by value in Machine create operations.
Machine	MachineStopForce	If true, the Provider shall support the “force” option on the stop and restart operations on Machines.
Machine	MachineStopForceDefault	If true, the Provider shall forcefully stop Machines if no other indication is provided. Otherwise, the Provider shall gracefully stop Machines.
Machine	RestoreFromImage	If true, the Provider supports restoring Machines from MachineImages that are not SNAPSHOT MachineImages.
Machine	UserData	If set, indicates which userData injection method shall be used by the Provider.
Machine	MachineAvailabilityLevel	If true, the Provider supports the notion of an availability level for the Machine Resource. The availability level and its value constraints are advertised as an extension attribute by the way of the Machine and MachineTemplate ResourceMetadata.
Credential	CredentialTemplateByValue	If true, the Provider shall support specifying CredentialTemplates by value in Credential create operations.
Volume	SharedVolumeSupport	If true, the Provider shall support that a single Volume Resource can be shared by multiple Machines.

Resource Name	Capability Name	Description
Volume	VolumeConfigByValue	If true, the Provider shall support specifying VolumeConfigurations by value in the Volume create operation. If true, the VolumeTemplateByValue capability shall have the value true.
Volume	VolumeImageByValue	If true, the Provider shall support specifying VolumeImages by value in the Volume create operation. If true, the VolumeTemplateByValue capability shall have the value true.
Volume	VolumeSnapshot	If true, the Provider shall support creating a new VolumeImage by referencing an existing Volume.
Volume	VolumeTemplateByValue	If true, the Provider shall support specifying the VolumeTemplates by value in Volume create operations.
Volume	VolumeAvailabilityLevel	If true, the Provider supports the notion of an availability level for the Volume Resource. The availability level and its value constraints are advertised as an extension attribute by the way of the Volume and VolumeTemplate ResourceMetadata.
Network	NetworkTemplateByValue	If true, the Provider shall support specifying Network Templates by value in Network create operations.
Network	DefaultInitialState	If this capability is set, unless otherwise provided (e.g., by a NetworkTemplate "initialState" attribute), the Provider shall set a new Network to this state value, assuming the value is compatible with the InitialStates capability, if set.
Network	InitialStates	If this capability is set, and if using a NetworkTemplate that has an “initialState” attribute, a Consumer shall use an initialState value from the set of values of this capability.
NetworkInterface	NetworkInterfaceTemplateByValue	If true, the Provider shall support specifying NetworkInterface Templates by value in NetworkInterface create operations.
NetworkInterface	DefaultInitialState	If this capability is set, unless otherwise provided (e.g., by a NetworkInterfaceTemplate "initialState" attribute), the Provider shall set a new NetworkInterface to this state value, assuming the value is compatible with the InitialStates capability, if set.
NetworkInterface	InitialStates	If this capability is set, and if using a NetworkInterfaceTemplate that has an “initialState” attribute, a Consumer shall use an initialState value from the set of values of this capability.
NetworkService	NetworkServiceTemplateByValue	If true, the Provider shall support specifying NetworkService Templates by value in NetworkService create operations.
NetworkService	DefaultInitialState	If this capability is set, unless otherwise provided (e.g., by a NetworkServiceTemplate "initialState" attribute), the Provider shall set a new NetworkService to this state value, assuming the value is compatible with the InitialStates capability, if set.
NetworkService	InitialStates	If this capability is set, and if using a NetworkServiceTemplate that has an “initialState” attribute, a Consumer shall use an initialState value from the set of values of this capability.
ProtocolEndpoint	ProtocolEndpointTemplateByValue	If true, the Provider shall support specifying ProtocolEndpoint Templates by value in ProtocolEndpoint create operations.
ProtocolEndpoint	DefaultInitialState	If this capability is set, unless otherwise provided (e.g., by a ProtocolEndpointTemplate "initialState" attribute), the Provider shall set a new ProtocolEndpoint to this state value, assuming the value is compatible with the InitialStates capability, if set.

Resource Name	Capability Name	Description
ProtocolEndpoint	InitialStates	If this capability is set, and if using a ProtocolEndpointTemplate that has an “initialState” attribute, a Consumer shall use an initialState value from the set of values of this capability.
ProtocolSegment	ProtocolSegmentTemplateByValue	If true, the Provider shall support specifying ProtocolSegment Templates by value in ProtocolSegment create operations.
ProtocolSegment	DefaultInitialState	If this capability is set, unless otherwise provided (e.g., by a ProtocolSegmentTemplate "initialState" attribute), the Provider shall set a new ProtocolSegment to this state value, assuming the value is compatible with the InitialStates capability, if set.
ProtocolSegment	InitialStates	If this capability is set, and if using a ProtocolSegmentTemplate that has an “initialState” attribute, a Consumer shall use an initialState value from the set of values of this capability.
Job	JobRetention	If set, the value of this capability shall indicate the minimum number of minutes a job shall be retained by the Provider before it is deleted.
Meter	MeterConfigByValue	If true, the Provider shall support specifying MeterConfigurations by value in Meter create operations.
Meter	MeterTemplateByValue	If true, the Provider shall support specifying MeterTemplates by value in Meter create operations.
EventLog	Linked	If true, the Provider shall delete EventLogs that are associated with Resources if the Resource is deleted.

The following examples show the `ResourceMetadata` for a `Machine` that advertises some of its capabilities:

JSON serialization:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/ResourceMetadata",
  "id": "http://example.com/types/Machine",
  "typeURI": "http://schemas.dmtf.org/cimi/2/Machine",
  "name": "Machine",
  "capabilities": [
    { "uri":
      "http://schemas.dmtf.org/cimi/2/capability/Machine/MachineConfigByValue",
      "value": true },
    { "uri":
      "http://schemas.dmtf.org/cimi/2/capability/Machine/MachineImageByValue",
      "value": true },
    { "uri":
      "http://schemas.dmtf.org/cimi/2/capability/Machine/DefaultInitialState",
      "value": "STARTED" }
  }
}
```

XML serialization:

```
<ResourceMetadata xmlns="http://schemas.dmtf.org/cimi/2">
  <id> http://example.org/types/Machine </id>
  <typeURI> http://schemas.dmtf.org/cimi/2/Machine </typeURI>
  <name> Machine </name>
  <capabilities>
    <capability
uri="http://schemas.dmtf.org/cimi/2/capability/Machine/MachineConfigByValue">
    true
    </capability>
    <capability
uri="http://schemas.dmtf.org/cimi/2/capability/Machine/MachineImageByValue">
    true
    </capability>
    <capability
uri="http://schemas.dmtf.org/cimi/2/capability/Machine/DefaultInitialState">
    STARTED
    </capability>
  </capabilities>
</ResourceMetadata>
```

5.11.2 ResourceMetadataCollection Resource

A `ResourceMetadataCollection` Resource represents the Collection of `ResourceMetadata` Resources within a Provider and follows the Collection pattern defined in clause 5.5.12. Note that modifications of the Resources within this Collection are typically reserved for administrator types of CIMI Consumers. This Resource shall be serialized as follows:

JSON serialization:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/ResourceMetadataCollection",
  "id": string,
  "count": number,
  "resourceMetadatas": [
    { "resourceURI": "http://schemas.dmtf.org/cimi/2/ResourceMetadata",
      "id": string,
      ... remaining ResourceMetadata attributes ...
    }, +
  ], ?
  "operations": [ { "rel": "add", "href": string } ? ]
  ...
}
```

XML serialization:

```
<Collection
    resourceURI="http://schemas.dmtf.org/cimi/2/ResourceMetadataCollection"
    xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <count> xs:integer </count>
  <resourceMetadatas>
    <ResourceMetadata>
      <id> xs:anyURI </id>
    ... remaining ResourceMetadata attributes ...
    </ResourceMetadata> *
  </resourceMetadatas>
  <operations>
    <operation rel="add" href="xs:anyURI"/> ?
  </operations>
  <xs:any>*
</Collection>
```

5.12 Cloud Entry Point

The Cloud Entry Point (`CloudEntryPoint` Resource) represents the entry point into the cloud defined by the CIMI Model. It provides a Consumer with a single address (URI) from which the Consumer can discover and access all Resources usable by this Consumer. A Cloud Provider may provide different CEPs to different Consumers. The Cloud Entry Point (CEP) implements a catalog of Resources, such as `Systems`, `SystemTemplates`, `Machines`, `MachineTemplates`, etc., that can be queried and browsed by the Consumer.

If a Consumer issues a read on the `CloudEntryPoint` Resource, the Provider shall return a `CloudEntryPoint` Resource that only catalogs Resources on which this Consumer is allowed to perform operations. **Error! Reference source not found.** describes the attributes for the `CloudEntryPoint` Resource.

If the delete operation is advertised on the CEP, deleting the `CloudEntryPoint` Resource is also deleting all referred Resources.

Table 8 – CloudEntryPoint attributes

Name	CloudEntryPoint	
Type URI	http://www.dmf.org/cimi/2/CloudEntryPoint	
Attribute	**Type**	**Description**
baseURI	*URI*	An absolute URI that references the "base URI" of the Provider. This URI shall be used to convert relative URIs to Resources within this Provider to absolute URIs. See the "URIs" clause of 5.5. **<u>Constraints:</u>** providerMandatory: true consumerMandatory: true mutable: false consumerWritable: false
resourceMetadata	*collection [Resource Metadata]*	A reference to `ResourceMetadata` Collection of this Cloud Entry Point. The Collection contains a description of the Resources supported by the Provider. If a Resource does not have any metadata, it shall not appear in this list, e.g., it has no constraints beyond what the CIMI specification defines nor does it have any extension attributes.
systems	*collection [System]*	A reference to the `SystemCollection` of this Cloud Entry Point.
systemTemplates	*collection [System Template]*	A reference to the `SystemTemplateCollection` of this CloudEntry Point.

Name	CloudEntryPoint	
Type URI	http://www.dmf.org/cimi/2/CloudEntryPoint	
Attribute	**Type**	**Description**
machines	*collection [Machine]*	A reference to the `MachineCollection` of this Cloud Entry Point.
machineTemplates	*collection [Machine Template]*	A reference to the `MachineTemplateCollection` of this Cloud Entry Point.
machineConfigs	*collection [Machine Configuration]*	A reference to the `MachineConfigurationCollection` of this Cloud Entry Point.
machineImages	*collection [Machine Image]*	A reference to the `MachineImageCollection` of this Cloud Entry Point.
credentials	*collection [Credential]*	A reference to the `CredentialCollection` of this Cloud Entry Point.
credentialTemplates	*collection [Credential Template]*	A reference to the `CredentialTemplateCollection` of this Cloud Entry Point.
volumes	*collection [Volume]*	A reference to the `VolumeCollection` of this Cloud Entry Point.
volumeTemplates	*collection [Volume Template]*	A reference to the `VolumeTemplateCollection` of this Cloud Entry Point.
volumeConfigs	*collection [Volume Configuration]*	A reference to the `VolumeConfigurationCollection` of this Cloud Entry Point.
volumeImages	*collection [Volume Image]*	A reference to the `VolumeImageCollection` of this Cloud Entry Point.
networks	*collection [Network]*	A reference to the `NetworkCollection` of this Cloud Entry Point.
networkTemplates	*collection [Network Template]*	A reference to the `NetworkTemplateCollection` of this Cloud Entry Point.
segments	*collection [Protocol Segment]*	A reference to the `ProtocolSegmentCollection` of this Cloud Entry Point.
segmentTemplates	*collection [Protocol Segment Template]*	A reference to the `ProtocolSegmentTemplateCollection` of this Cloud Entry Point.
endpoints	*collection [Protocol Endpoint]*	A reference to the `ProtocolEndpointCollection` of this Cloud Entry Point.
endpointTemplates	*collection [Protocol Endpoint Templates]*	A reference to the `ProtocolEndpointTemplateCollection` of this Cloud Entry Point.
interfaces	*collection [Network Interface]*	A reference to the `NetworkInterfaceCollection` of this Cloud Entry Point.
interfaceTemplates	*collection [Network Interface Templates]*	A reference to the `NetworkInterfaceTemplateCollection` of this Cloud Entry Point.
networkServices	*collection [Network*	A reference to the `NetworkServiceCollection` of this Cloud Entry Point.

Name	CloudEntryPoint	
Type URI	http://www.dmf.org/cimi/2/CloudEntryPoint	
Attribute	**Type**	**Description**
	Service]	
networkServiceTemplates	*collection [Network Service Template]*	A reference to the `NetworkServiceTemplateCollection` of this Cloud Entry Point.
jobs	*collection [Job]*	A reference to the `JobsCollection` of this Cloud Entry Point.
meters	*collection [Meter]*	A reference to the `MeterCollection` of this Cloud Entry Point.
meterTemplates	*collection [Meter Template]*	A reference to the `MeterTemplateCollection` of this Cloud Entry Point.
meterConfigs	*collection [Meter Configuration]*	A reference to the `MeterConfigurationCollection` of this Cloud Entry Point.
eventLogs	*collection [EventLog]*	A reference to the `EventLogCollection` of this Cloud Entry Point.
eventLogTemplates	*collection [EventLog Template]*	A reference to the `EventLogTemplateCollection` of this Cloud Entry Point.

Every above attribute of the `CloudEntryPoint` Resource has the following constraints by default (unless overridden per attribute):

providerMandatory: false
consumerMandatory: false
mutable: true
consumerWritable: true

Each of the Collections mentioned in **Error! Reference source not found.** are defined within the related Resource definition clauses. For example, the `MachineCollection` Resource is defined in clause 5.14.2 as part of the Machine-related Resources.When implementing or using `CloudEntryPoint`, Providers and Consumers shall adhere to the syntax and semantics of its attributes as described in **Error! Reference source not found.** as well as in the tables describing embedded Resources or related Collections. 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:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/CloudEntryPoint",
  "id": string,
  "name": string, ?
  "description": string, ?
  "created": string, ?
  "updated": string, ?
  "properties": { string: string, + }, ?
  "baseURI": string,
  "resourceMetadata": { "href": string }, ?
  "systems": { "href": string }, ?
  "systemTemplates": { "href": string }, ?
  "machines": { "href": string }, ?
  "machineTemplates": { "href": string }, ?
  "machineConfigs": { "href": string }, ?
  "machineImages": { "href": string }, ?
  "credentials": { "href" string }, ?
  "credentialTemplates": { "href" string }, ?
  "volumes": { "href": string }, ?
  "volumeTemplates": { "href": string }, ?
  "volumeConfigs": { "href": string }, ?
  "volumeImages": { "href": string }, ?
  "networks": { "href": string }, ?
  "networkTemplates": { "href": string }, ?
  "segments": { "href": string }, ?
  "segmentTemplates": { "href": string }, ?
  "endpoints": { "href": string }, ?
  "endpointTemplates": { "href": string }, ?
  "interfaces": { "href": string }, ?
  "interfaceTemplates": { "href": string }, ?
  "networkServices": { "href": string }, ?
  "networkServiceTemplates": { "href": string }, ?
  "jobs": { "href": string }, ?
  "meters": { "href": string }, ?
  "meterTemplates": { "href": string }, ?
  "meterConfigs": { "href": string }, ?
  "eventLogs": { "href": string }, ?
  "eventLogTemplates": { "href": string }, ?
  "operations": [
    { "rel": "edit", "href": string } ?
  ] ?
  ...
}
```

XML media type: application/xml

XML serialization:

```
<CloudEntryPoint 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> ?
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties>
  <baseURI> xs:anyURI </baseURI>
  <resourceMetadata href="xs:anyURI"/> ?
  <systems href="xs:anyURI"/> ?
  <systemTemplates href="xs:anyURI"/> ?
  <machines href="xs:anyURI"/> ?
  <machineTemplates href="xs:anyURI"/> ?
  <machineConfigs href="xs:anyURI"/> ?
  <machineImages href="xs:anyURI"/> ?
  <credentials href="xs:anyURI"/> ?
  <credentialTemplates href="xs:anyURI"/> ?
  <volumes href="xs:anyURI"/> ?
  <volumeTemplates href="xs:anyURI"/> ?
  <volumeConfigs href="xs:anyURI"/> ?
  <volumeImages href="xs:anyURI"/> ?
  <networks href="xs:anyURI"/> ?
  <networkTemplates href="xs:anyURI"/> ?
  <segments href="xs:anyURI"/> ?
  <segmentTemplates href="xs:anyURI"/> ?
  <endpoints href="xs:anyURI"/> ?
  <endpointTemplates href="xs:anyURI"/> ?
  <interfaces href="xs:anyURI"/> ?
  <interfaceTemplates href="xs:anyURI"/> ?
  <networkServices href="xs:anyURI"/> ?
  <networkServiceTemplates href="xs:anyURI"/> ?
  <jobs href="xs:anyURI"/> ?
  <meters href="xs:anyURI"/> ?
  <meterTemplates href="xs:anyURI"/> ?
  <meterConfigs href="xs:anyURI"/> ?
  <eventLogs href="xs:anyURI"/> ?
  <eventLogTemplates href="xs:anyURI"/> ?
  <operations>
    <operation rel="edit" href="xs:anyURI"/> *
  </operations>
  <xs:any>*
</CloudEntryPoint>
```

5.12.1 Operations

This Resource supports the Read and Update operations.

5.13 System Resources and relationships

5.13.1 System

A `System` is a realized Resource that consists of one or more `Networks`, `Volumes`, `Machines`, (and others) that could be connected and associated with each other. A `System` can be created from the interpretation of a `SystemTemplate`. A `System` can be operated and managed as a single Resource and usually forms a stack of service. For example, a shopping cart system consists of machines for Web servers and databases, network addresses for public access, and volumes for database files. A `System` has several "top-level" attributes that are Collections of references to Resources of various types. Each one of these Collections shall contain references to Resource items of the related type that are components of the System. Each one of these System components may be either:

- a *child component* of the the System (see 5.10.2).
- an *associated component* of the System..

By default, all Resources that are created as the result of a System creation are also children components of the `System`. Some components of a System may pre-exist to the System – e.g. they would be referred to by the SystemTemplate used to create that System. Such component Resources are associated components of the System

An example of associated component in a System, is of a Network created independently from the System, directly by POSTing to the `networks` CEP collection. A Consumer may then want the System to reuse that Network as a component while keeping the Network managed separately from the System, in particular not to be deleted when the System is deleted. Such a Network may still be inserted in the `networks` System collection as an associated component, while having its `parent` attribute referring to the CEP as originally set. Alternatively, the Network could be made a child component of the System by setting its `parent` attribute to the System's `networks` collection Resource.

Note:

- A Resource component of a `System` may in turn use some other Resources that are not component of this `System`, e.g., a `Machine` in a `System` can use a `Volume` that is neither component of the Machine, nor a component of the System.

Error! Reference source not found. describes the System attributes.

Table 9 – System attributes

Name	System	
Type URI	http://schemas.dmtf.org/cimi/2/System	
Attribute	**Type**	**Description**
state	*string*	The operational state of the System. Allowed values are: (See 5.14.1.) **CREATING**: The System is in the process of being created. **STARTING/STARTED/STOPPING/STOPPED/PAUSING/PAUSED/SUSPENDING/SUSPENDED**: The `System` shall be in one of these states if all the `Machines` referenced by the `System` are in that state. See clause 5.14.1 for the list of available actions based on the state of a `Machine`. Such transitional states may just indicate that all `Machines` in a `System` are undergoing the same operation (e.g., "start"), without the `System` being actually operated on (e.g., no "start" done at `System` level). An actual operation on a `System` may be traced by querying the "job" entity. **MIXED**: The `System` shall be in this state if either no `Machines` are referenced

Name	System	
Type URI	http://schemas.dmtf.org/cimi/2/System	
Attribute	**Type**	**Description**
		by this System or Machines referenced by this System are in varying states. Such varying states are likely to occur when an operation is in progress on a System, resulting in transitions of its Machine states toward a new common state (e.g., STOPPED, STARTED) but at a different pace, or sequentially one after the other. **DELETING**: The System is in the process of being deleted. **ERROR**: The Provider has detected an error in the System. The operations that result in transitions to the above defined states are defined in clause 5.13.1.2.
systems	*collection [System]*	A list of references to nested Systems that are components of this System.
machines	*collection [Machine]*	A list of references to Machines that are components of this System.
credentials	*collection [Credential]*	A list of references to Credentials that are components of this System.
volumes	*collection [Volume]*	A list of references Volumes that are components of this System.
networks	*collection [Network]*	A list of references to Network that are components of this System.
networkServices	*collection [Network Service]*	A reference to the NetworkServiceCollection that are components of this System.
services	*Collection [SystemService]*	A list of references to SystemService Resources that represent services supported by this System.
meters	*collection [Meter]*	A list of references to Meters monitored for this System, with component semantics. Note that these Meters are for the System and not for any individual component in the System.
eventLog	*ref*	A reference to the EventLog of this System. Note that this EventLog is for the System and not for any individual component in the System.

When implementing or using System, Providers and Consumers shall adhere to the syntax and semantics of its attributes as described in **Error! Reference source not found.** as well as in the tables describing embedded Resources or related Collections.

5.13.1.1 Attributes of type Collection

The following clause describes the Collection Resources components of Systems.

5.13.1.1.1 systems Collection

The Resource type for each item of this Collection is “System”. There is no accessory attribute for the items in this Collection, therefore, it is a basic System Collection, the serialization of which follows the rules in 5.5.12. See the SystemCollection Resource clause.

5.13.1.1.2 machines Collection

The Resource type for each item of this Collection is "`Machine`". There is no accessory attribute for the items in this Collection, therefore, it is a basic Machine Collection (serialized as described in 5.5.12). See the MachineCollection Resource clause.

5.13.1.1.3 credentials Collection

The Resource type for each item of this Collection is "`Credential`". There is no accessory attribute for the items in this Collection, therefore, it is a basic `Credential` Collection (serialized as described in 5.5.12). See the CredentialCollection Resource clause.

5.13.1.1.4 volumes Collection

The Resource type for each item of this Collection is "`Volume`". There is no accessory attribute for the items in this Collection, therefore, it is a basic `Volume` Collection (serialized as described in 5.5.12). See the VolumeCollection Resource clause.

5.13.1.1.5 networks Collection

The Resource type for each item of this Collection is "`Network`". There is no accessory attribute for the items in this Collection, therefore, it is a basic `NetworkCollection` Resource as described in clause.5.16.2

5.13.1.1.6 networkServices Collection

The Resource type for each item of this Collection is "`NetworkService`". There is no accessory attribute for the items in this Collection, therefore, it is a basic `NetworkServiceCollection` as described in clause 5.16.18.

5.13.1.1.7 meters Collection

The Resource type for each item of this Collection is "`Meter`" as defined in clause 5.17.3. There is no accessory attribute for the items in this Collection, thereforem it is a basic `Meter` Collection (serialized as described in 5.5.12). See the MeterCollection Resource clause.

5.13.1.2 Operations

The `System` Resource supports the Read, Update, and Delete operations. Create is supported through the `SystemCollection` Resource.

The following custom operations are also defined:

start/stop/restart/pause/suspend

/link@rel: `http://schemas.dmtf.org/cimi/2/action/xxx`

Where "xxx" is either "`start`", "`stop`", "`restart`", "`pause`", or "`suspend`".

This operation shall recursively perform the requested operation on each component of the `System` (`Machine` or sub-`System`). Note that not all `Machines` need to be in the same state for this operation to be available and the impact of this operation varies depending on the component's current state; see clause 5.14.1.2 for more details about performing operations on `Machines`. If the operation fails for a `Machine`, that `Machine` shall not be affected by the operation.

export

/link@rel: `http://schemas.dmtf.org/cimi/2/action/export`

This operation shall export a `System` along with all Resources component of or used by this System. If an export package exists at that URI, it is updated with the values of the `System` and any component management Resources. Otherwise, a new export package is created at that URI with a Media Type as specified by the "format" parameter. Other formats may be used if supported, but are not specified by this standard.

Input parameters:

1) "format" - type: string - optional
2) Indicates the Media Type of the exported data. If not present, the default value shall be "application/ovf."
3)
4) "destination" - type: URI - optional
5) Indicates the location to where the exported data is placed. If not present, the HTTP response Location header shall contain the URL to the exported data. Based on the specific protocol specified within the URI, the Consumer might need to provide additional information (such as credentials) in the "properties" field. In the case of HTTP, a PUT shall be used to place the data at the specified location.

Output parameters: None.

HTTP protocol

To export a `System`, a POST is sent to the "http://schemas.dmtf.org/cimi/2/action/export" URI of the `System` where the HTTP request body shall be as described below.

JSON media type: application/json

JSON serialization:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/Action",
  "action": "http://schemas.dmtf.org/cimi/2/action/export",
  "format": string, ?
  "destination": string, ?
  "properties": { string: string, + } ?
  ...
}
```

XML media type: application/xml

XML serialization

```
<Action xmlns="http://schemas.dmtf.org/cimi/2">
  <action>http://schemas.dmtf.org/cimi/2/action/export</action>
  <format> xs:string </format> ?
  <destination> xs:anyURI </destination> ?
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties>
  <xs:any>*
```

```
</Action>
```

5.13.2 SystemCollection Resource

A `SystemCollection` Resource represents a Collection of `System` Resources and follows the Collection pattern defined in clause 5.5.12. Operations

NOTE The "add" operation requires that a SystemTemplate be used (see 4.2.1.1).

Resources created during the process of creating a `System` shall be components of the `System` (see 5.13.1). For example, a `componentDescriptor` that references a `MachineTemplate`, and within that `MachineTemplate` is a reference to a `VolumeTemplate`, results in a reference to the new `Machine` being added to the `System.machines` attribute and a reference to the new `Volume` being added to the `System.volumes` attribute. However, if this `MachineTemplate` refers to an existing `Volume`, this `Volume` shall not be added to the top-level `System` attributes.

The following custom operations are also defined:

import

/link@rel:`http://schemas.dmtf.org/cimi/2/action/import`

This operation shall import a `System`. Not only is a `System` created, but `Machines`, `Volumes`, and `Networks` and possibly recursive `Systems` and their components may also be created corresponding to imported descriptor entries. More detail about this process is in ANNEX A.

1) Input parameters:"source" - type: URI - mandatory
2) Indicates the location from which the imported data is retrieved. Based on the specific protocol specified within the URI, the Consumer might need to provide additional information (such as credentials) in the "properties" field.

Output parameters: None.

HTTP protocol

To import a `System`, a POST is sent to the "http://schemas.dmtf.org/cimi/2/action/import" URI of the System `Collection` where the HTTP request body shall be as described below.

JSON media type: application/json

JSON serialization:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/Action",
  "action": "http://schemas.dmtf.org/cimi/2/action/import",
  "source": string, ?
  "properties": { string: string, + } ?
  ...
}
```

XML media type: application/xml

XML serialization

```
<Action xmlns="http://schemas.dmtf.org/cimi/2">
  <action> http://schemas.dmtf.org/cimi/2/action/import </action>
  <source> xs:anyURI </source> ?
  <properties>
```

```
    <property key="xs:string"> xs:string </property> *
  </properties>
  <xs:any>*
</Action>
```

5.13.3 SystemService Resource

A SystemService Resource represents some management service for all or a subset of the Resources in a System. A SystemService Resource can define diverse types of management services and holds:

(a) Topology information about the service: a list of the Resources concerned by this management service, e.g. lists of Machines and Volumes subject to disaster recovery policy.
(b) Service-specific parameters: configuration data for the service itself.

System components may be listed under more than one `SystemService` Resources. For example, a `Machine` may be under a recovery service, while also participating into an autoscaling service.

Some examples of common services are:

- HighReliability service
- DisasterRecovery service
- Backup service
- Autoscaling service

Table 10 - SystemService attributes

Name	SystemService	
Type URI	http://schemas.dmtf.org/cimi/2/SystemService	
Attribute	**Type**	**Description**
serviceType	*URI*	Unique URI identifying this particular service. It shall be of the form: http://schemas.dmtf.org/cimi/2/SystemService/<servicename> where <servicename is the end of the path, possibly a subpath.
machines	*Collection[Machine]*	A reference to the list of references to `Machines` that are managed under this `SystemService`. The Resource item type may be a variant of Machine in case accessory attributes are added to the collection. This Resource items in this Collection are not child components of the `SystemService` Resource: deleting the `SystemService` shall not cause the deletion of the referred Machines.
volumes	*Collection[Volume]*	A reference to the list of references to `Volumes` that are managed under this `SystemService`. The Resource item type may be a variant of `Volume` in case accessory attributes are added to the collection. This Resource items in this Collection are not child components of the `SystemService` Resource: deleting the `SystemService` shall not cause the deletion of the referred `Volumes`.
systems	*collection [System]*	A reference to the list of references to `Systems` or sub-`Systems` that are managed under this `SystemService`. The Resource item type may be a variant of `System`in case accessory attributes are added to the collection. This Resource items in this Collection are not child components of the `SystemService` Resource: deleting the `SystemService` shall not cause the deletion of the referred `Systems.`
parameters	*map*	A list of attributes that are specific to this SystemService, i.e. associated with a particular `ServiceType` value.

5.13.3.1 HighReliability service Resource

This service allows for a System to recover from the failures of its Machines; the service intervenes when the Machine stops working - typically the System does not receive the Machine heartbeat anymore. This service protects from hardware and software failures, i.e. the failure of the hardware node executing the machine, or the case of a software process causing a segment violation that stops the OS services.

Table 11 – SystemService attributes for HighReliability service

Name	SystemService	
Type URI	http://schemas.dmtf.org/cimi/2/SystemService	
Attribute	**Type**	**Description**
serviceType	*URI*	http://schemas.dmtf.org/cimi/2/SystemService/highreliability/active or http://schemas.dmtf.org/cimi/2/SystemService/highreliability/passive
machines	*Collection[Recoverable Machine]*	A reference to the collection of `Machines` in the `System` that are managed under this `SystemService`, meaning these benefit from recovery service. Adding a `Machine` reference to this collection means that the `Machine` becomes managed under this `SystemService`. • If the serviceType is ending with "/highreliability/active": Then each one of the listed `Machines` has a backup `Machine.` In case of failure the backup `Machine` (referred to by the `recoverableMachine` collection item) shall take over. . • If the serviceType is ending with "/highreliability/passive": Then each one of the listed `Machines` has an up-to-date `MachineImage.` In case of failure the backup `Machine` is created from the `MachineImage` and shall replace the failed `Machine.` , This Resource items in this Collection are not components of the `SystemService` Resource: deleting the `SystemService` does not cause the deletion of the referred Machines. The details of the `SystemService` behavior (e.g. failover detection, etc.) depends on the Provider's implementation, and can be controlled by additional parameters in the next attribute.
parameters	*map*	

name	***type***	**value**
networkServices	*collection [Network Service]*	A reference to the `NetworkServiceCollection` within the System that support this `SystemService`.
heartbeat	Integer	Heartbeat frequency, in term of millisecs between an heartbeat and the next.
replicationType	String	The kind of Machine replication status (it does not refer to the Volume Resource) allowable values are: **synchronous**, **asynchronous**, **none**, (same Machine, but not status alignment in order to allow the recovery in case just the status could cause failure**)** **onlyAtClusterCreation**

Name	SystemService	
Type URI	http://schemas.dmtf.org/cimi/2/SystemService	
Attribute	**Type**	**Description**
RPO	Integer	Recovery Point Objective (duration in minutes) in case of asynchronous replica of the disks.

5.13.3.1.1 `RecoverableMachine` Collection

The referred Resource type for each item of this Collection is "`Machine`". However because there are accessory attributes, this is not a basic but an enhanced Machine Collection. The accessory attribute is defined in Table 18:

Table 12 – RecoverableMachine accessory attributes

Name	RecoverableMachine	
Type URI		
Attribute	**Type**	**Description**
backupmachine	*Ref*	An additional reference to the backup Machine in the same System, that supports the Machine referenced by this collection item.

5.13.3.1.2 Operations

The `HighReliability` SystemService Resource supports the Read, Update, and Delete operations. Create is supported through the SystemService Collection Resource.

Adding a machine to the collection (see the `addRM` operation) implies that a backup Machine shall be created and the `backupmachine` attribute shall be assigned to this copy (even if it is not an running Machine, but only a "passive" copy ready to be executed in case of failure). The way the backup copy is created depends on the Provider implementation, it is expected that an image of the recoverable machine is taken and from this image a new machine is created.

If the Consumer also gives the backupmachine reference as input parameter, it is assumed that the backup machine is that referenced machine and no new backup machines shall be created.

A backup machine may also be added as part of the list of recovertable machines (i.e. in the "machines" collection of the Systerm service). This amounts to defining a daisy-chain of two (or more) backup machines for the original (primary) recoverable machine subject to the system service.

The following custom operations are also defined on this SystemService Resource:

forceSync

/link@rel: `http://schemas.dmtf.org/cimi/2/action/forceSync`

This operation shall synchronize the state of a node onto its backup node, regardless of the scheduled synchronization time as dictated by the recovery policies.

The result of this operation depends on the Provider implementation and on the status of the backup Machine; typically it has effect when the backup Machine is obtained by an image copy of the recoverable Machine.

Input parameters: "node" (primary node) type: ref - mandatory

Output parameters: None.

swapBackup

/link@rel: `http://schemas.dmtf.org/cimi/2/action/swapBackup`

This operation shall swap a Machine and its backup Machine – i.e. replace the Machine with its backup and vice-versa.

Some Providers can choose to not make available this operation, not allowing the Consumer to choose which backup node turn in primary one.

Input parameters:"node" - type: ref - mandatory
A reference to the Machine to be replaced by its backup

Output parameters: None.

addRM

/link@rel: `http://schemas.dmtf.org/cimi/2/action/addRM`

This operation adds a recoverable Machine (or RM) to the collection of recoverable Machines under this service. It adds the reference of the Machine to the `machines` collection of recoverable Machines, and optionally a reference to the backup Machine (accessory attribute “backupmachine”)..

Input parameters:"node" (Machine to be added to the service) - type: ref – mandatory, “backup” (Machine to be used as backup) - type: ref – optional ,
Output parameters: None.

removeRM

/link@rel: `http://schemas.dmtf.org/cimi/2/action/removeRM`

This operation removes a recoverable Machine (or RM) from the collection of recoverable Machines under this service. It removes the reference of the Machine from the `machines` collection of recoverable Machines, and discards the backup Machine.

Input parameters:"node" (Machine to be removed from the service) - type: ref – mandatory,
Output parameters: None.

5.13.3.2 DisasterRecovery service Resource

This service allows for a System to recover from a data center failure – by maintaining a remote, up-to-date image of the System.

Unlike the `HighReliability` service, which enables to define advanced recovery techniques for different error typologies, the `DisasterRecovery` service intervenes in the specific case of a data center failure and only implements the mechanism to re-start crashed resources on a remote data center.

On a data center failure occurrence, where other advanced approaches fail, this service guarantees resources’ restoration, although some service-downtime will occur, i.e. there should be no expectation from the customers that transition from one data center to another is “transparent”.

Typically the `DisasterRecovery` can be offered by default for every Machine, though some Providers could activate it as an additional feature to be explicitly requested by the Consumer, or more often could allow the consumer to chose the location of the remote datacenter; in such cases it is possible to define a `DisasterRecovery` service Resource.

The attribues for the DisasterRecovery system service Resource are:

Table 13 – SystemService attributes for DisasterRecovery service

Name	SystemService	
Type URI	http://schemas.dmtf.org/cimi/2/SystemService	
Attribute	**Type**	**Description**
serviceType	*URI*	http://schemas.dmtf.org/cimi/2/SystemService/disasterrecovery/
machines	*Collection[Machine]*	A reference to the collection of `Machines` in the `System` that are managed under this `SystemService`, meaning these benefit from recovery service. Adding a `Machine` reference to this collection means that the `Machine` becomes managed under this `SystemService`. This Resource items in this Collection are not components of the `SystemService` Resource: deleting the `SystemService` does not cause the deletion of the referred Machines. The details of the `SystemService` behavior (e.g. failover detection, etc.) depends on the Provider's implementation.
parameters	*map*	

name	***type***	**value**
backupData Center	*URI*	Identity of the backup data center or Cloud to be used as a backup.
backupCEP	*ref*	Reference to the CEP in the backup DC under which the recovery resources are to be provisioned.
networkServi ces	*collection [Network Service]*	A reference to the `NetworkServiceCollection` within the System that support this `SystemService`.

5.13.3.2.1 Operations

The `DisasterRecovery` SystemService Resource supports the Read, Update, and Delete operations. Create is supported through the SystemService Collection Resource.

addRM

/link@rel: `http://schemas.dmtf.org/cimi/2/action/addRM`

This operation adds a recoverable Machine (or RM) to the collection of recoverable Machines under this service. It adds the reference of the Machine to the `machines` collection of recoverable Machines.

Input parameters:"node" (Machine to be added to the service) - type: ref – mandatory,
Output parameters: None.

removeRM

/link@rel: `http://schemas.dmtf.org/cimi/2/action/removeRM`

This operation removes a recoverable Machine (or RM) from the collection of recoverable Machines under this service. It removes the reference of the Machine from the `machines` collection of recoverable Machines..

Input parameters:"node" (Machine to be removed from the service) - type: ref – mandatory,
Output parameters: None.

5.13.4 SystemTemplate Resource

The `SystemTemplate` Resource contains the set of individual descriptors that are necessary to create or associate the components of a `System`. In practice, the Provider interprets the set of component descriptors as a set of creation (or association) operations to be executed in an order compatible with the dependencies (e.g., attachments or references between components) that are expressed between these components.

A `SystemTemplate` may include symbolic component references in the descriptors, used to express links between components of the resulting System. A component reference uses the "name" of the target (referred) component. For example, `<volume href="#newVolume"/>` would reference a `Volume` named "newVolume." The reference name – #newVolume – is replaced by the actual Resource URL in the instantiated System.

Table 19 describes the `SystemTemplate` attributes.

Table 14 – SystemTemplate attributes

Name	SystemTemplate	
Type URI	http://schemas.dmtf.org/cimi/2/SystemTemplate	
Attribute	**Type**	**Description**
component Descriptors	*component Descriptor*[]	The list of component descriptors describing the components of a `System` instance realized from this `SystemTemplate`. For each component descriptor, the corresponding component is either created when a `System` instance is created (i.e. a child component), or simply associated with the system if it already exists. • In case of a child component: The component descriptor refers to a Template (either by reference or by value), and may also provide additional metadata (name, description, properties). The creation order of components is not specified in `SystemTemplate`; in particular the order of the component descriptors in this array is not meaningful in terms of creation order. • In case of an existing Resource to be added as an associated component of the System: The component descriptor refers directly to the existing Resource.

Name	*componentDescriptor*	
Data	**Type**	**Description**
name	*string*	The value of the "name" attribute that is associated with a `System` component created from this component descriptor. Note: This name is not to be confused with the name that may be present in the component Template – e.g., a `MachineTemplate` – from which this component is instantiated.
description	*string*	The value of the "description" attribute that is associated with a `System` component created from this component descriptor.
properties	*map*	The key/value pairs that is associated with a `System` component created from this component descriptor.
type	*URI*	The TypeURI of the component to be created from this component descriptor, e.g., for a `Machine`: http://schemas.dmtf.org/cimi/2/Machine

Name	SystemTemplate			
Type URI	http://schemas.dmtf.org/cimi/2/SystemTemplate			
Attribute	**Type**	**Description**		

		<component Resource>	<any>	The exact name of this attribute varies depending on the type of Resource being created or added,
				This attribute shall contain either:
				• A Template that is provided inline. Such an embedded Template may contain component references, each one of which shall resolve to the URI of a component with same name once created from this `SystemTemplate`. In such a case, the attribute name is same as the Template type name, with first letter lower case. (e.g. machineTemplate).
				• A reference to an externally defined Template. Some attribute name/value pairs may be added inside the componentTemplate element to override similar attributes in the referred Template (as described in 4.2.1.1). This example shows how component references can be added to an external Template. The attribute name is same as the Template type name, with first letter lower case. (e.g. machineTemplate).
				Example (JSON):
				`"machineTemplate": {`
				`    "href":`
				`"http://example.com/machineTemplates/72000",`
				`     "credential": { "href": "#MyCredential"`
				`}`
				
				`  }`
				Note: The "credential" attribute in this example assumes that there is another componentDescriptor item named "MyCredential" of type "Credential" in the SystemTemplate. It shall set or override similar attribute in the referred MachineTemplate if instantiating the Machine component.
				• A reference to an existing Resource to become associated component of the System. The attribute name is same as the Resource type name, with first letter lower case (e.g. "machine).

		quantity	*integer*	The number of component instances to be created from this component descriptor, if a template. By default, this number is equal to 1. If the value is 2 or more, the actual name assigned to each instance is the "name" value concatenated with a sequential number (e.g., if name="mymachine", and quantity=3, the names are: mymachine1, mymachine2, mymachine3.)
serviceDescriptors	*serviceDescriptor[]*	The list of service descriptors for the services to be supported by a `System` instance realized from this `SystemTemplate`. For each service descriptor, the corresponding `SystemService` is created when a `System` instance is created. The names of the System components subject to the service are listed using the symbolic component reference notation previously described ("**#**<name>").		
		Name	*serviceDescriptor*	
		Data	**Type**	**Description**
		name	*string*	The value of the "name" attribute that is associated with a `SystemService` instance created from this service descriptor.

Name	SystemTemplate			
Type URI	http://schemas.dmtf.org/cimi/2/SystemTemplate			
Attribute	**Type**	**Description**		
		description	*string*	The value of the "description" attribute that is associated with a `SystemService` instance created from this service descriptor.
		properties	*map*	The key/value pairs that is associated with a `SystemService` instance created from this service descriptor.
		serviceType	*URI*	The serviceType of the service to be created from this service descriptor, e.g., for a `SystemService` of type "DisasterRecovery": `http://schemas.dmtf.org/cimi/2/SystemService/disasterrecovery`
		parameters	*map*	This is where additional service-specific attributes are listed (see section 5.13.6).
meter Templates	*Meter Templates[]*	A list of references to `MeterTemplates` that shall be used to create and connect a set of new `Meters` to the new `System`. Note that the attributes of the `MeterTemplate` may be specified rather than a reference to an existing `MeterTemplate` Resource.		
eventLog Template	*ref*	A reference to an `EventLogTemplate` that shall be used to create and connect a new `EventLog` to the new `System`. Note that the attributes of the `EventLogTemplate` may be specified rather than a reference to an existing `EventLogTemplate` Resource.		
import Image	*URI*	If the Template is the result of an import – e.g., of an OVF package - this attribute should be used. If present, it shall reference the import source (e.g., OVF package) used to create this Template.		
genResourc eMetadata	*ref*	A reference to a `ResourceMetadata` that shall be associated with every System generated from this Template.		

When implementing or using `SystemTemplate`, Providers and Consumers shall adhere to the syntax and semantics of its attributes as described in Table 19 as well as in the tables describing embedded Resources or related Collections.

5.13.4.1 Operations

This Resource supports the Read, Update, and Delete operations. Create is supported through the `SystemTemplateCollection` Resource.

The following custom operations are also defined:

export

/link@rel: `http://schemas.dmtf.org/cimi/2/action/export`

This operation shall export a `SystemTemplate` along with all its component Resources as well as the used Resources that are listed in its top-level Collections . If an export package exists at that URI, it is updated with the values of the `SystemTemplate` and any component management Resources.

Otherwise a new export package is created at that URI with a Media Type as specified by the "format" parameter. Other formats may be used if supported, but are not specified by this standard.

Input parameters:

1) "format" - type: string - optional
2) Indicates the Media Type of the exported data. If not present, the default value shall be "application/ovf."
3) "destination" - type: URI - optional
4) Indicates the location to where the exported data is placed. If not present, the HTTP response Location header shall contain the URL to the exported data. Based on the specific protocol specified within the URI, the Consumer might need to provide additional information (such as credentials) in the "properties" field. In the case of HTTP, a PUT shall be used to place the data at the specified location.

Output parameters: None.

HTTP protocol

To export a `SystemTemplate`, a POST is sent to the "http://schemas.dmtf.org/cimi/2/action/export" URI of the `SystemTemplate` where the HTTP request body shall be as described below.

JSON media type: application/json

JSON serialization:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/Action",
  "action": "http://schemas.dmtf.org/cimi/2/action/export",
  "format": string, ?
  "destination": string, ?
  "properties": { string: string, + } ?
  ...
}
```

XML media type: application/xml

XML serialization

```
<Action xmlns="http://schemas.dmtf.org/cimi/2">
  <action> http://schemas.dmtf.org/cimi/2/action/export </action>
  <format> xs:string </format> ?
  <destination> xs:anyURI </destination> ?
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties> ?
  <xs:any>*
</Action>
```

5.13.5 SystemTemplateCollection Resource

A `SystemTemplateCollection` Resource represents the Collection of `SystemTemplate` Resources within a Provider and follows the Collection pattern defined in clause 5.5.12.

5.13.5.1 Operations

The following custom operations are defined:

import

/link@rel: `http://schemas.dmtf.org/cimi/2/action/import`

This operation shall import a `SystemTemplate`. Not only is a `SystemTemplate` created, but `MachineTemplates`, `VolumeTemplates`, and `NetworkTemplates` and possibly recursive `SystemTemplates` and their components may also be created, corresponding to imported descriptor entries. More detail about this process is in ANNEX A.

Input parameters:

1) "source" - type: URI - mandatory
2) Indicates the location from which the imported data is retrieved. Based on the specific protocol specified within the URI, the Consumer might need to provide additional information (such as credentials) in the "properties" field.

Output parameters: None.

HTTP protocol

To import a `SystemTemplate`, a POST is sent to the "http://schemas.dmtf.org/cimi/2/action/import" URI of the `SystemTemplateCollection` where the HTTP request body shall be as described below.

JSON media type: application/json

JSON serialization:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/Action",
  "action": "http://schemas.dmtf.org/cimi/2/action/import",
  "source": string, ?
  "properties": { string: string, + } ?
  ...
}
```

XML media type: application/xml

XML serialization

```
<Action xmlns="http://schemas.dmtf.org/cimi/2">
  <action> http://schemas.dmtf.org/cimi/2/action/import </action>
  <source> xs:anyURI </source> ?
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties> ?
  <xs:any>*
</Action>
```

5.13.6 Service-specific Descriptor attributes

This section defines the additional attributes specific to each service type that need be added to a serviceDescriptor for this service type in the SystemTemplate.

5.13.6.1 Parameters for the HighReliability service type

Service type: `http://schemas.dmtf.org/cimi/2/SystemService/highreliability`

Table 15 – Additional parameters for HighReliability service

Service type	`highreliability`	
Attribute	**Type**	**Description**
machines	*String[]*	Symbolic references to the `Machine` components in the `System` that are subject to the service. Uses the symbolic component reference notation previously described ("**#**<name>").
network	*string*	Symbolic reference to the `Network` Resource in the `System` that enables this service. The `Network` shall provide the necessary connections between Machines to support this Service..
heartbeat	Integer	Heartbeat frequency, in term of millisecs between an heartbeat and the next.
replicationType	String	The kind of disk replication data (it does not refer to the Volume Resource) allowable values are: **synchronous**, **asynchronous**, **none**, **onlyAtClusterCreation**
RPO	Integer	Recovery Point Objective (duration in minutes) in case of asynchronous replica of the disks.

5.14 Machine Resources and relationships

5.14.1 Machine

An instantiated compute Resource that encapsulates both CPU and Memory. Table 16 describes the `Machine` attributes.

Table 16 – Machine attributes

Name	Machine	
Type URI	http://schemas.dmtf.org/cimi/2/Machine	
Attribute	**Type**	**Description**
state	*string*	The operational state of the Machine. Allowed values are: **CREATING**: The `Machine` is in the process of being created. **STARTING**: The `Machine` is in the process of being started. **STARTED**: The `Machine` is available and ready for use. **STOPPING**: The `Machine` is in the process of being stopped. **STOPPED**: This value is the virtual equivalent of powering off a physical `Machine`. There is no saved CPU or memory state. Clause 0 defines the initial state of a `Machine`. **PAUSING**: The `Machine` in the process of being PAUSED.

Name	Machine	
Type URI	http://schemas.dmtf.org/cimi/2/Machine	
Attribute	**Type**	**Description**
		PAUSED: In this state the Machine and its virtual resources remain instantiated and resources remain allocated, similar to the "STARTED" state, but the Machine and its virtual resources are not enabled to perform tasks. This is equivalent to a "stand-by" state. **SUSPENDING**: The Machine is in the process of being suspended. **SUSPENDED**: In this state the Machine and its virtual resources are stored on non-volatile storage. The Machine and its resources are not enabled to perform tasks. **CAPTURING:** If the Machine is undergoing the "capture" operation its state may be set to "CAPTURING". If some operations that were accepted by the Machine before the capture are no longer available during the capture, the Machine shall be in state "CAPTURING. **RESTORING:** The Machine is in the process of being restored from a MachineImage. **DELETING**: The Machine is in the process of being deleted. **ERROR**: The Provider has detected an error in the Machine. **FAILED**: the Machine is not operational due to some error condition and in accordance to the Provider's policies it is considered *failed*. This state calls for a recovery procedure, if any. The operations that result in transitions to the above defined states are defined in clause 5.14.1.2.
cpu	*integer*	The amount of CPU that this Machine has.
memory	*integer*	The size of the memory (RAM) in kibibytes allocated to this Machine. If this value is increased, it implies that the Machine is allocated more RAM, and vice versa if the value is decreased.
disks	*collection [Disk]*	A reference to the list of disks (local storage) that are part of the Machine. Adding an element to this list creates a disk. The Disk Resources are components of the Machine. Note: The Disk Resource type is defined in clause 5.14.1.1.1.
cpuArch	*string*	The CPU architecture that is supported by Machines created by using this configuration. Allowed values are: **68000**, **Alpha**, **ARM**, **Itanium**, **MIPS**, **PA_RISC**, **POWER**, **PowerPC**, **x86**, **x86_64**, **z/Architecture**, **SPARC**. Providers may define additional values.
cpuSpeed	*integer*	The approximate CPU speed of this Machine - in megahertz.
volumes	*collection [located Volume]*	A reference to the list of references to Volumes that are connected to this Machine. Adding a Volume to this list means that the Machine has some access to the data on the Volume. Removing a Volume from this list means that the Machine no longer has access to the data on the Volume. Note: . This Collection has the semantics of usage of the Volumes by the Machine (deleting the Machine does not cause the deletion of the referred Volumes). It is defined in clause 5.14.1.1.2.

Name	Machine	
Type URI	http://schemas.dmtf.org/cimi/2/Machine	
Attribute	**Type**	**Description**
interfaces	*collection [Network Interface]*	A reference to a list of references to NetworkInterfaces on this Machine. Each NetworkInterface Resource is a component of the Machine Resource. Each NetworkInterface instance represents an association between the Machine and a Network. NetworkInterfaces are defined in clause 5.16.13.
latestSnapshot	*ref*	A reference to the SNAPSHOT representing the latest state captured for this Machine (either most recent Snapshot or the last Snapshot reverted to). **<u>Constraints:</u>** **Provider:** support optional; mutable **Consumer:** support optional; read-only
snapshots	*collection [MachineImage]*	A reference to the list of references to the MachineImages of type SNAPSHOT taken of this Machine. This Collection has the semantics of usage of SNAPSHOT MachineImages by the Machine (The deletion of the Machine does not cause the deletion of the referred Snapshots.)
meters	*collection [Meter]*	A reference to the list of Meters monitored for this Machine.
eventLog	*ref*	A reference to the EventLog of this Machine.

When implementing or using Machine, Providers and Consumers shall adhere to the syntax and semantics of its attributes as described in Table 16, as well as in the tables describing embedded Resources or related Collections.

5.14.1.1 Collections

The following clause describes the Collection Resources components of Machines.

5.14.1.1.1 Disk Collection

The Resource type for each item of this Collection is "Disk", defined in Table 17:

Table 17 – Disk attributes

Name	Disk	
Type URI	http://schemas.dmtf.org/cimi/2/Disk	
Attribute	**Type**	**Description**
capacity	*integer*	The initial capacity, in kilobytes, of the disk.
initialLocation	*string*	Operating System-specific location (path) in its namespace where this disk first appears. After deployment, Consumers may consider moving the location of this Disk.. Support of this attribute indicates that the Provider can report this information back to the Consumer.

5.14.1.1.2 volumes Collection

The referred Resource type for each item of this Collection is “Volume”. However because there is an accessory attribute (initialLocation), this is not a basic but an enhanced Volume Collection. The name “locatedVolume” is used to define the type of each Collection item. The accessory attribute is defined in Table 18:

Table 18 – `locatedVolume` accessory attributes

Name	locatedVolume	
Type URI	http://schemas.dmtf.org/cimi/2/locatedVolume	
Attribute	**Type**	**Description**
initialLocation	*string*	Operating System-specific location (path) in its namespace where this `Volume` first appears. Note, once deployed, Consumers might move the location of this `Volume`. Support of this attribute indicates that the Provider can report this information back to the Consumer.

The `resourceURI` attribute value for the Collection of locatedVolume items is: `http://schemas.dmtf.org/cimi/2/locatedVolumeCollection`.

5.14.1.1.3 interfaces Collection

The Resource type for each item of this Collection is "`NetworkInterface`", defined in clasue 5.16.13. The Collection is a basic `NetworkInterfaceCollection` as described in clause 5.16.14.

5.14.1.1.4 snapshots Collection

The Resource type for each item of this Collection is "`MachineImage`". It is a basic `MachineImage` Collection. Its serialization is described in the `MachineImageCollection` Resource clause.

5.14.1.1.5 meters Collection

The Resource type for each item of this Collection is "`Meter`" as defined in clause 5.17.3. There is no accessory attribute for the items in this Collection, therefore it is a basic `Meter` Collection (serialized as described in 5.5.12). See the MeterCollection Resource clause.

5.14.1.2 Operations

This Resource supports the Read, Update, and Delete operations. Create is supported through the MachineCollection Resource.

The following custom operations are also defined:

start

/link@rel: `http://schemas.dmtf.org/cimi/2/action/start`

This operation shall start a `Machine`.

Input parameters: None.

Output parameters: None.

During the processing of this operation, the `Machine` shall be in the "STARTING" state.

Upon successful completion of this operation, the `Machine` shall be in the "STARTED" state.

If a `Machine` is in the "STOPPED" state, starting it shall be the virtual equivalent of powering on a physical machine. There is no restored CPU or Memory state, so the guest OS typically performs boot or installation tasks.

If the `Machine` was in the "SUSPENDED" or "PAUSED" state, starting it shall have the effect of resuming it.

HTTP protocol

To start a `Machine`, a POST is sent to the "http://schemas.dmtf.org/cimi/2/action/start" URI of the `Machine` where the HTTP request body shall be as described below.

JSON media type: application/json

JSON serialization:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/Action",
  "action": "http://schemas.dmtf.org/cimi/2/action/start",
  "properties": { string: string, + } ?
  ...
}
```

XML media type: application/xml

XML serialization

```
<Action xmlns="http://schemas.dmtf.org/cimi/2">
  <action> http://schemas.dmtf.org/cimi/2/action/start </action>
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties>
  <xs:any>*
</Action>
```

Upon successful processing of the request, the HTTP response body may be empty.

stop

/link@rel: `http://schemas.dmtf.org/cimi/2/action/stop`

This operation shall stop a `Machine`.

Input parameters:

1) "force" - type: boolean - optional
2) A flag to indicate whether the Provider shall simulate a power off condition (force=true) or shall simulate a shutdown operation that allows applications to save their state and the file system to be made consistent (force=false). Inclusion of this parameter by Consumers is optional and if not specified, the Provider may choose either mechanism. Providers are encouraged to advertise this choice by the way of the MachineStopForceDefault capability.

Output parameters: None.

During the processing of this operation, the `Machine` shall be in the "STOPPING" state.

Upon successful completion of this operation, the `Machine` shall be in the "STOPPED" state. Stopping a `Machine` with force=true shall be the virtual equivalent of powering off a physical machine. There is no saved CPU or Memory state. Stopping a `Machine` with force=false shall result in a machine with consistent file systems.

A Consumer may reissue a stop operation if the state is STOPPING, perhaps with force=true, but Providers shall not issue a force=true stop operation on their own.

HTTP protocol

To stop a `Machine`, a POST is sent to the "http://schemas.dmtf.org/cimi/2/action/stop" URI of the `Machine` where the HTTP request body shall be as described below.

JSON media type: application/json

JSON serialization:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/Action",
  "action": "http://schemas.dmtf.org/cimi/2/action/stop",
  "force": boolean, ?
  "properties": { string: string, + } ?
  ...
}
```

XML media type: application/xml

XML serialization

```
<Action xmlns="http://schemas.dmtf.org/cimi/2">
  <action> http://schemas.dmtf.org/cimi/2/action/stop </action>
  <force> xs:boolean </force> ?
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties>
  <xs:any>*
</Action>
```

Upon successful processing of the request, the HTTP response body may be empty.

restart

/link@rel: `http://schemas.dmtf.org/cimi/2/action/restart`

This operation shall restart a `Machine`. If the `Machine` is in the "STARTED" state, this operation shall have the effect of executing the "stop" and then "start" operations. If the `Machine` is in the "STOPPED" state, this operation shall have the effect of executing the "start" operation.

Input parameters:

1) "force" - type: boolean - optional
2) A flag to indicate whether the Provider shall simulate a power off condition (force=true) or shall simulate a shutdown operation that allows applications to save their state and the file system to be made consistent (force=false). Inclusion of this parameter by Consumers is optional and if not specified, the Provider may choose either mechanism. Providers are encouraged to advertise this choice by the way of the MachineStopForceDefault capability.

Output parameters: None.

During the processing of this operation, the `Machine` shall be in the “STOPPING” and/or “STARTING” states, as appropriate depending on its initial state.

Upon successful completion of this operation, the `Machine` shall be in the "STARTED" state. Restarting a `Machine` shall be the virtual equivalent of powering off, and then powering on a physical machine. There is no restored CPU or Memory state, so the guest OS typically performs boot or installation tasks.

HTTP protocol

To restart a `Machine`, a POST is sent to the "http://schemas.dmtf.org/cimi/2/action/restart" URI of the `Machine` where the HTTP request body shall be as described below.

JSON media type: application/json

JSON serialization:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/Action",
  "action": "http://schemas.dmtf.org/cimi/2/action/restart",
  "force": boolean, ?
  "properties": { string: string, + } ?
  ...
}
```

XML media type: application/xml

XML serialization

```
<Action xmlns="http://schemas.dmtf.org/cimi/2">
  <action> http://schemas.dmtf.org/cimi/2/action/restart </action>
  <force> xs:boolean </force> ?
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties>
  <xs:any>*
</Action>
```

Upon successful processing of the request, the HTTP response body may be empty.

pause

/link@rel: `http://schemas.dmtf.org/cimi/2/action/pause`

This operation shall pause a `Machine`.

Input parameters: None.

Output parameters: None.

During the processing of this operation, the `Machine` shall be in the "PAUSING" state.

Upon successful completion of this operation, the `Machine` shall be in the "PAUSED" state. Pausing a `Machine` shall keep the `Machine` and its resources instantiated, but the `Machine` shall not be available to perform any tasks. The current state of the CPU and Memory shall be retained in volatile memory.

HTTP protocol

To pause a `Machine`, a POST is sent to the "http://schemas.dmtf.org/cimi/2/action.pause" URI of the `Machine` where the HTTP request body shall be as described below.

JSON media type: application/json

JSON serialization:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/Action",
  "action": "http://schemas.dmtf.org/cimi/2/action/pause",
  "properties": { string: string, + } ?
  ...
}
```

XML media type: application/xml

XML serialization

```
<Action xmlns="http://schemas.dmtf.org/cimi/2">
  <action> http://schemas.dmtf.org/cimi/2/action/pause </action>
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties>
  <xs:any>*
</Action>
```

Upon successful processing of the request, the HTTP response body may be empty.

suspend

/link@rel: `http://schemas.dmtf.org/cimi/2/action/suspend`

This operation shall suspend a `Machine`.

Input parameters: None.

Output parameters: None.

During the processing of this operation, the `Machine` shall be in the "SUSPENDING" state.

Upon successful completion of this operation, the `Machine` shall be in the "SUSPENDED" state. Suspending a `Machine` shall keep the `Machine` and its resources instantiated, but the `Machine` shall not be available to perform any tasks. The current state of the CPU and Memory shall be retained in non-volatile memory.

HTTP protocol

To suspend a `Machine`, a POST is sent to the "http://schemas.dmtf.org/cimi/2/action/suspend" URI of the `Machine` where the HTTP request body shall be as described below.

JSON media type: application/json

JSON serialization:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/Action",
  "action": "http://schemas.dmtf.org/cimi/2/action/suspend",
  "properties": { string: string, + } ?
  ...
}
```

XML media type: application/xml

XML serialization

```
<Action xmlns="http://schemas.dmtf.org/cimi/2">
  <action> http://schemas.dmtf.org/cimi/2/action/suspend </action>
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties>
  <xs:any>*
</Action>
```

Upon successful processing of the request, the HTTP response body may be empty.

capture

/link@rel: `http://schemas.dmtf.org/cimi/2/action/capture`

This operation shall create a new `MachineImage` from an existing `Machine`. This operation is defined within the `MachineImage` Resource; see 5.14.7.1 for more details. Note that while this operation is performed against a `MachineImage`, its presence in the `Machine` serialization is used to advertise support for the operation.

Snapshotting a Machine

/link@rel: `http://schemas.dmtf.org/cimi/2/action/snapshot`

This operation shall create a new SNAPSHOT `MachineImage` from an existing `Machine`. This operation is defined within the `MachineImage` Resource; see 5.14.7.1 for more details. Note that while this operation is performed against a `MachineImage`, its presence in the `Machine` serialization is used to advertise support for the operation.

Restoring a Machine

/link@rel: `http://schemas.dmtf.org/cimi/2/action/restore`

This operation shall restore a `Machine` from a previously created `MachineImage`.

Input parameters:

1) "image" - type: URI - mandatory
2) A reference to the Machine Image.

Output parameters: None.

During the processing of this operation, the `Machine` shall be in the "RESTORING" state.

Upon successful completion of this operation, the `Machine` shall be in the same state as the state specified in the `MachineImage`, if specified. See 0 for more details.

Note that Providers can indicate support for restoring from non-SNAPSHOT `MachineImages` by the way of the Machine "RestoreFromImage" capability. If the RestoreFromImage capability is not supported, and the restore operation is supported, the restore operation can only restore from a SNAPSHOT `MachineImage`.

HTTP protocol

To restore a `Machine`, a POST is sent to the "http://schemas.dmtf.org/cimi/2/action/restore" URI of the `Machine` where the HTTP request body shall be as described below.

JSON media type: application/json

JSON serialization:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/Action",
  "action": "http://schemas.dmtf.org/cimi/2/action/restore",
  "image": { "href": string },
  "properties": { string: string, + } ?
  ...
}
```

XML media type: application/xml

XML serialization

```
<Action xmlns="http://schemas.dmtf.org/cimi/2">
  <action> http://schemas.dmtf.org/cimi/2/action/restore </action>
  <image href="xs:anyURI"/>
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties>
  <xs:any>*
</Action>
```

Where the "image" URI is a reference to the `MachineImage` to be used.

Upon successful processing of the request, the HTTP response body may be empty.

connectvolume

/link@rel: `http://schemas.dmtf.org/cimi/2/action/connectvolume`

This operation shall start a `Machine`.

Input parameters: Volume reference, initialLocation, Credentials, ….

Output parameters: None.

HTTP protocol

To connect a Volume to a `Machine`, a POST is sent to the "http://schemas.dmtf.org/cimi/2/action/`connectvolume`" URI of the `Machine` where the HTTP request body shall be as described below.

JSON media type: application/json

JSON serialization:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/Action",
  "action": "http://schemas.dmtf.org/cimi/2/action/connectvolume",
  “volume”: { "href": string },
  “initialLocation”: string,
  “credentials”: { "href": string },
  "properties": { string: string, + } ?
  ...
}
```

XML media type: application/xml

XML serialization

```
<Action xmlns="http://schemas.dmtf.org/cimi/2">
  <action> http://schemas.dmtf.org/cimi/2/action/connectvolume</action>
  <volume href="xs:anyURI"/>
  <initialLocation>xs:string</initialLocation>
  <credentials href="xs:anyURI"/>
  <action> http://schemas.dmtf.org/cimi/2/action/connectvolume</action>
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties>
  <xs:any>*
</Action>
```

Upon successful processing of the request, the HTTP response body may be empty.

5.14.2 MachineCollection Resource

A `MachineCollection` Resource represents the Collection of `Machine` Resources within a Provider and follows the Collection pattern defined in clause 5.5.12. Operations

NOTE The "add" operation requires that a MachineTemplate be used (see 4.2.1.1).

Upon successful processing of the "add" operation, unless otherwise specified by the way of the `MachineTemplate` "initialState" attribute, the state of the new `Machine` shall be the value of the DefaultInitialState capability, if defined. If no DefaultInitialState capability is defined, the default value shall be "STOPPED." The semantics of "initialState” shall be equivalent to the Provider issuing the appropriate actions against the new `Machine` to move it into that state. Note that this controls the actions of the hypervisor and the state of the resources within the `Machine` (e.g., the operating system) are also influenced by the data within the `MachineImage` used to create the new `Machine`. For example, if a new `Machine's` initialState is "STARTED" and a SNAPSHOT `MachineImage` was used to create the new `Machine`, the `Machine` would not be "booted" but rather resume executing from the saved state in the `MachineImage`.

If a Provider is unable to change the state of the new `Machine` to the appropriate "initialState" (either as specified by the `MachineTemplate` or as implied by the previous stated rules), the `Machine` creation shall fail.

If a Provider is unable to create the new `Machine` due to invalid or inconsistent credentials in the `MachineTemplate`, the `Machine` creation process shall fail. If any credentials are included in the `MachineTemplate`, they shall be part of the new `Machine` regardless of the type of `MachineImage` used.

5.14.3 MachineTemplate

A MachineTemplate represents the set of metadata and instructions used in the creation of a Machine. Table 19 describes the MachineTemplate attributes.

Table 19 – MachineTemplate attributes

Name	MachineTemplate	
Type URI	http://schemas.dmtf.org/cimi/2/MachineTemplate	
Attribute	**Type**	**Description**
initialState	*string*	The initial state of the new Machine. Possible values include the non-transient states as specified by the Machine "state" attribute (e.g., STARTED, STOPPED) and are determined by the actions supported by the Provider. Providers should advertise the list of available values through the Machine's "initialStates" capability.
machineConfig	*ref*	A reference to the MachineConfiguration that is used to create a Machine from this MachineTemplate. Note that the attributes of the MachineConfiguration may be specified rather than a reference to an existing MachineConfiguration Resource.
machineImage	*ref*	A reference to the MachineImage that is used to create a Machine from this MachineTemplate.
credential	*ref*	A reference to the Credential that is used to create the initial login credentials for the new Machine. Note that the attributes of the Credential may be specified rather than a reference to an existing Credential Resource.
volumes	*volume[]*	A list of structures, each containing a reference to an existing Volume and potentially describing aspects of the way that the given Volume is to be connected to the Machine during its creation from this MachineTemplate. Each volume structure has the following attributes:

Name	*volume*	
Attribute	**Type**	**Description**
initialLocation	*string*	An Operating System-specific location (path) in its namespace where the Volume appears. Support of this attribute indicates that the Provider allows for Consumers to choose where the Volume appears.
credential	*ref*	Credential for accessing the Volume to be connected (if necessary).
volume	*ref*	Reference to the Volume that is connected.

Attribute	**Type**	**Description**
volumeTemplates	*volumeTemplate[]*	A list of structures, each containing a reference to a VolumeTemplate from which a Volume is created and connected to the Machine resulting from this MachineTemplate. Each structure can potentially also include aspects of the way in which each created Volume is connected to the created Machine. . Credentials associated with the new Volume are same as the Credentials for this Machine. If the Machine is created as part of a System creation, the Volumes created from these Templates are considered as part of that System without the need for these VolumeTemplates to also be listed in the volumeTemplates attribute of the relevant SystemTemplate. If the same VolumeTemplate reference is listed in both the volumeTemplates

Name	MachineTemplate	
Type URI	http://schemas.dmtf.org/cimi/2/MachineTemplate	
Attribute	**Type**	**Description**
		attribute of a `SystemTemplate` and in the volumeTemplates attribute of a `MachineTemplate` component of that `SystemTemplate`, this means that multiple, distinct `Volume` instances are created as part of the overall `System` creation. Each volumeTemplate structure has the following attributes:

Name	*volumeTemplate*	
Attribute	**Type**	**Description**
initialLocation	*string*	An Operating System-specific location (path) in its namespace where the Volume appears. Support of this attribute indicates that the Provider allows for Consumers to choose where the Volume appears.
volumeTemplate	*ref*	Reference to the `VolumeTemplate` that is used to create a new `Volume`. Note that the attributes of the `VolumeTemplate` may be specified rather than a reference to an existing `VolumeTemplate` Resource.

interfaceTemplates	*Network Interface Template[]*	A list of references to `NetworkInterfaceTemplates` that shall be used to create a new set of `NetworkInterface` Resources for the new `Machine`. Note that the attributes of a `NetworkInterfaceTemplate` may be given instead of a reference to an existing `NetworkInterfaceTemplate` Resource.
userData	*string*	A Base64 encoded string whose decoded version is to be injected into Machines created by using this Template. See the discussion of injection of user-defined data below.
meterTemplates	*meterTemplates[]*	A list of references to `MeterTemplates` that shall be used to create and connect a set of new `Meters` to the new `Machine`. Note that the attributes of the `MeterTemplate` may be specified rather than a reference to an existing `MeterTemplate` Resource.
eventLogTemplate	*ref*	A reference to an `EventLogTemplate` that shall be used to create and connect a new `EventLog` to the new `Machine`. Note that the attributes of the `EventLogTemplate` may be specified rather than a reference to an existing `EventLogTemplate` Resource.
genResourceMetadata	*ref*	A reference to a `ResourceMetadata` that shall be associated with every Machine generated from this Template.

When implementing or using `MachineTemplate`, Providers and Consumers shall adhere to the syntax and semantics of its attributes as described in Table 19, as well as in the tables describing embedded Resources or related Collections.

Injection of user-defined data

To simplify the customization of individual `Machines`, it is possible to pass arbitrary data into the new `Machine` by using the userData parameter. The value of this parameter shall be the Base64-encoded payload. The Provider shall arrange for this data to be available from inside the `Machine` by using one of the following methods:

1. *Metadata server*: The data can be retrieved from within the instance by using an HTTP GET request to http://169.254.169.254/cimi/latest/user-data.
2. *Disk*: The `Machine` has access to a Disk with an ISO 9660 file system on it. The data can be found in a file at <*location*>/cimi/user-data.
3. *Image modification*: The Provider modifies the root file system of the machine image just before launching the `Machine`. In UNIX-like operating systems, the data can be found in the file /var/lib/cimi/user-data.

It is strongly recommended that Providers implement a `metadata server`, or, failing that, injection by the way of `Disk`, as `image modification` is brittle and may not work for every operating system in use. The Provider shall indicate which of these three methods is supported with the `Machine` 'UserData' capability in the `ResourceMetadata` for `Machines`. The value for this feature shall be one of metadata, disk, or imgmod, corresponding to the three methods listed above.

The Provider shall preserve this data across restarts of the `Machine`. The data is the Base64-decoded version of the data that was passed into the `MachineCreate` request.

5.14.3.1 Operations

This Resource supports the Read, Update, and Delete operations. Create is supported through the MachineTemplateCollection Resource.

5.14.4 MachineTemplateCollection Resource

A `MachineTemplateCollection` Resource represents the Collection of `MachineTemplate` Resources within a Provider and follows the Collection pattern defined in clause 5.5.12. Operations

This Resource supports the Read and Update operations. Creation of new `MachineTemplate` Resources is supported by the way of a POST to the "add" operation's URI as described in clause 4.2.1.1.

5.14.5 MachineConfiguration Resource

The `MachineConfiguration` Resource represents the set of configuration values that define the (virtual) hardware resources of a to-be-realized Machine Instance. `MachineConfigurations` are created by Providers and may, at the Providers discretion, be created by Consumers.

Table 20 describes the `MachineConfiguration` attributes.

Table 20 – MachineConfiguration attributes

Name	MachineConfiguration	
Type URI	http://schemas.dmtf.org/cimi/2/MachineConfiguration	
Attribute	**Type**	**Description**
cpu	*integer*	The amount of CPU that a `Machine` realized from this configuration.
memory	*integer*	The amount of RAM, in kibibytes, that a `Machine` realized from this configuration.
disks	disk[]	A list of structures, each containing the attributes defining the disks to be created for the `Machine` instantiated with this `MachineConfiguration` Resource. The disks are local storage to the Machine. Each disks attribute has the following sub-attributes:

Name	MachineConfiguration	
Type URI	http://schemas.dmtf.org/cimi/2/MachineConfiguration	
Attribute	**Type**	**Description**

Name	*disk*	
Attribute	**Type**	**Description**
capacity	*integer*	The initial capacity, in kilobytes, of the disk described by this attribute.
format	*string*	The format/type of this disk (e.g., ext4, NTFS).
initialLocation	*string*	An Operating System-specific location (path) in its namespace where this Disk first appears. After creation of a Machine, Consumers may change the location of this Disk.

Attribute	**Type**	**Description**
cpuArch	string	The CPU architecture that is supported by Machines created by using this configuration. Allowed values are: **68000**, **Alpha**, **ARM**, **Itanium**, **MIPS**, **PA_RISC**, **POWER**, **PowerPC**, **x86**, **x86_64**, **z/Architecture**, **SPARC**. Providers may define additional values.
cpuSpeed	*integer*	The approximate CPU speed of this Machine in megahertz.

NOTE The disk attributes "format" does not appear on Machine Resources because after the Machine is created, the user of the Machine is able modify this attribute of a disk, possibly without the Provider's knowledge. Therefore these attributes might not be an aspect of the Machine that the Provider can reliably manage.

5.14.5.1 Operations

This Resource supports the Read, Update, and Delete operations. Create is supported through the MachineConfigurationCollection Resource.

5.14.6 MachineConfigurationCollection Resource

A MachineConfigurationCollection Resource represents the Collection of MachineConfiguration Resources within a Provider and follows the Collection pattern defined in clause 5.5.12.

5.14.6.1 Operations

This Resource supports the Read and Update operations. Creation of new MachineConfiguration Resources is supported by the way of a POST to the "add" operation's URI as described in clause 4.2.1.1.

5.14.7 MachineImage Resource

This Resource represents the information necessary for hardware virtualized Resources to create a Machine Instance; it contains configuration data such as startup instructions, including possible combinations of the following items, depending on the “type” of MachineImage created:

- The software image (i.e., a copy of an installed Machine), that is to be instantiated on the disk and other virtual resources. The image can be a snapshot that consists of disk images plus memory and other resource state information.
- Installation software, which, when executed on the hardware (virtual) resources, builds the machine instance.
- Both a disk image and a set of software and parameters to install new components not included in the original disk image.

Table 21 describes the MachineImage attributes.

Table 21 – MachineImage attributes

Name	MachineImage	
Type URI	http://schemas.dmtf.org/cimi/2/MachineImage	
Attribute	**Type**	**Description**
state	*string*	The operational state of the MachineImage. Allowed values are: **CREATING**: The MachineImage is in the process of being created. **AVAILABLE**: The MachineImage is available and ready for use. Unless otherwise specified, the MachineImage shall initially be in this state after successful creation. **DELETING**: The MachineImage is in the process of being deleted. **ERROR**: The Provider has detected an error in the MachineImage. The operations that result in transitions to the above defined states are defined in clause 5.14.7.1
type	*string*	The type of MachineImage that is represented by this Resource. This specification defines the following values: **IMAGE**: This type represents the persisted data of a stopped Machine. Unlike "snapshots", it does not contain any runtime information. If this value is used, the "relatedImage" attribute shall not be present. **SNAPSHOT**: This type represents the persisted data of a Machine. If the Machine was not in a stopped state when his Image was created, it also contains runtime information. If this value is used, the "relatedImage" attribute shall reference the most recently created (or reverted to) snapshot Image for that Machine, which allows for easy discovery of the "previous" snapshot. The "relatedImage" attribute shall not be set by Consumers. **PARTIAL_SNAPSHOT**: This type follows the same semantics as the "SNAPSHOT" MachineImage except that it contains just the changes (deltas) made to the Machine based on the referenced "relatedImage" MachineImage rather than a complete representation of the Machine. If a MachineImage is deleted, the following semantics shall apply: • Any "SNAPSHOT" MachineImages that have a "relatedImage" value that references the deleted MachineImage shall have that value changed to the "relatedImage" attribute of the delete MachineImage. • Any "PARTIAL_SNAPSHOT" MachineImages that have a "relatedImage" value that references the deleted MachineImage shall also be deleted. This detail applies recursively to any subsequent "PARTIAL_SNAPSHOT" MachineImages as well.
imageLocation	*URI*	A reference to the location of the binary data that makes up this image.
relatedImage	*ref*	A reference to another MachineImage Resource that is related to this one. The specific meaning of this value varies depending on the type of MachineImage.

5.14.7.1 Operations

This Resource supports the Read, Update, and Delete operations. Create is supported through the MachineImageCollection Resource.

If creating a new MachineImage, the representation of the new MachineImage may include a reference in the "imageLocation" attribute. Providers shall inspect this reference (most likely by the way of an HTTP HEAD) to determine if any special processing is required. This specification defines the following additional steps that Providers shall take depending on the type of Resource being referenced:

`http://schemas.dmtf.org/cimi/2/Machine`

If the "imageLocation" is a reference to a `Machine`, the Provider shall create a new `MachineImage` based on the `Machine` being referenced. The machine is captured or snapshotted, depending on whether the request was sent to the "http://schemas.dmtf.org/cimi/2/action/capture" or the "http://schemas.dmtf.org/cimi/2/action/snapshot" URI of the Machine. However the resulting resource, although linked to the Machine from which it was originated, shall be a MachineImage for all purposes and can be used for creating new machines.

If creating a SNAPSHOT and upon completion of the create operation, the `MachineImage's` "imageLocation" attribute shall not reference the `Machine` (as the `Machine` might change over time), but instead it shall reference (or contain the data of) the static representation of the `Machine`. Additionally, the referenced `Machine's MachineSnapshotCollection` shall be updated to include a reference to this newly created SNAPSHOT `MachineImage` Resource. If the Machine is unable to accept operations at any point while it is being captured to create the MachineImage, the Machine shall be in state “CAPTURING”.

5.14.8 MachineImageCollection Resource

A `MachineImageCollection` Resource represents the Collection of `MachineImage` Resources within a Provider and follows the Collection pattern defined in clause 5.5.12.

5.14.8.1 Operations

This Resource supports the Read and Update operations. Creation of new `MachineImage` Resources is supported by the way of a POST to the "add" operation's URI as described in clause 4.2.1.1, where the request body and the way it is processed are described in clause 5.14.7.1.

5.14.9 Credential Resource

A `Credential` Resource contains the information required to create the initial administrative superuser of a newly created `Machine` or to represent the credentials needed to perform some operation. Due to the variation between operating systems and Providers, this specification does not mandate one particular set of attributes that all implementations need to support. However, Providers are expected to extend this Resource with additional attributes to meet their requirements.

For example, a Provider might extend this Resource with username and password attributes, which would then be the login information for new `Machines`. These extension attributes would appear as siblings to the common attributes like "name"' and "description."

Table 22 describes the `Credential` attributes.

Table 22 – Credential attributes

Name	Credential	
Type URI	http://schemas.dmtf.org/cimi/2/Credential	
Attribute	**Type**	**Description**
TBD		The exact set of attributes is determined by the Provider.

Some common extension attributes that Providers might use include:

Table 23 – UserName/Password attributes

Attribute	Type	Description
userName	*string*	Initial superuser's user name.
password	*string*	Initial superuser's password.

Table 24 – Public key attributes

Attribute	Type	Description
key	*byte[]*	The digit of the public key for the initial superuser.

5.14.9.1 When implementing or using `Credential`, Providers and Consumers shall adhere to the syntax and semantics of its attributes as described in the above table, as well as in the table describing related Collections. **Operations**

This Resource supports the Read, Update, and Delete operations. Create is supported through the `CredentialCollection` Resource.

5.14.10 CredentialCollection Resource

A `CredentialCollection` Resource represents the Collection of `Credential` Resources within a Provider and follows the Collection pattern defined in clause 5.5.12.

5.14.10.1 Operations

NOTE The "add" operation requires that a CredentialTemplate be used (see 4.2.1.1).

5.14.11 CredentialTemplate Resource

This Resource captures the configuration values for realizing a `Credential` Resource. A `CredentialTemplate` may be used to create multiple `Credentials`. Table 25 describes the `CredentialTemplate` attributes.

Table 25 – CredentialTemplate attributes

Name	CredentialTemplate	
Type URI	http://schemas.dmtf.org/cimi/2/CredentialTemplate	
Attribute	**Type**	**Description**
TBD		The exact set of attributes is determined by the provider.

When implementing or using `CredentialTemplate`, Providers and Consumers shall adhere to the syntax and semantics of its attributes as described in Table 25 as well as in the table describing related Collections.

5.14.11.1 Operations

This Resource supports the Read, Update, and Delete operations. Create is supported through the `CredentialTemplateCollection` Resource.

5.14.12 CredentialTemplateCollection Resource

A `CredentialTemplateCollection` Resource represents the Collection of `CredentialTemplate` Resources within a Provider and follows the Collection pattern defined in clause 5.5.12.

5.14.12.1 Operations

This Resource supports the Read and Update operations. Creation of new `CredentialTemplate` Resources is supported by the way of a POST to the "add" operation's URI as described in clause 4.2.1.1.

5.15 Volume Resources and relationships

5.15.1 Volume

A `Volume` represents storage at either the block or the file-system level. `Volumes` can be connected to `Machines`. Once connected, `Volumes` can be accessed by processes on that `Machine`. Table 26 describes the `Volume` attributes.

Table 26 – Volume attributes

Name	Volume	
Type URI	http://schemas.dmtf.org/cimi/2/Volume	
Attribute	**Type**	**Description**
state	*string*	The operational state of the `Volume`. Allowed values are: **CREATING**: The `Volume` is in the process of being created. **AVAILABLE**: The `Volume` is available and ready for use. Unless otherwise specified, the `Volume` shall be in this state initially after successful creation. **CAPTURING**: The `Volume` is in the process of being captured (snapshotted) into a new `VolumeImage`. **RESTORING**: The Volume is in the process of being restored. **DELETING**: The `Volume` is in the process of being deleted. **ERROR**: The Provider has detected an error in the `Volume`. **<u>The operations that result in transitions to the above defined states are defined in clause 5.15.1.2</u>**
type	*URI*	A URI that indicates the type of `Volume` to be created. This specification defines the following URI: **http://schemas.dmtf.org/cimi/2/mapped**: Indicates a `Volume` that shall be used for shared storage that might be available to multiple `Machines`, but which does not require an explicit mount operation from within the guest operating system. Additional values may be defined. If certain types of `Volumes` require additional data, it is expected that this Resource is extended. For example, a "sharedFileSystem" type might require additional networking information and credentials to be specified.
capacity	*integer*	The maximum size, if limited, of the `Volume` in kilobytes. If this value is increased, the `Volume` can contain more data. Decreasing this value may require evaluations.
bootable	*boolean*	This property indicates whether this `Volume` is bootable.
images	*collection [Volume Image]*	A reference to the list of references to `VolumeImages` that represent snapshots taken from the `Volume`. Note: . This Collection has the semantics of usage of `VolumeImages` by the `Volume` (deleting the `Volume` does not cause the deletion of the referred `VolumeImages`)
meters	*collection [Meter]*	A reference to the list of `Meters` monitored for this `Volume`.
eventLog	*ref*	A reference to the `EventLog` of this `Volume`.

When implementing or using `Volume`, Providers and Consumers shall adhere to the syntax and semantics of its attributes as described in the above table as well as in the tables describing embedded Resources or related Collections.

5.15.1.1 Collections

The following clauses describe the Collection Resources owned by `Volumes`.

5.15.1.1.1 images Collection

The Resource type for each item of this Collection is "`VolumeImage`". There is no accessory attribute for the items in this Collection, therefore it is a basic `VolumeImage` Collection (serialized as described in 5.5.12).

See the `VolumeImageCollection` Resource clause.

NOTE Previous versions of this specification included an "add" operation on this Resource. It is now deprecated in favor of creating a new `VolumeImage` with the imageLocation attribute pointing to the `Volume` to be captured.

5.15.1.1.2 meters Collection

The Resource type for each item of this Collection is "`Meter`" as defined in clause 5.17.3. There is no accessory attribute for the items in this Collection, therefore it is a basic `Meter` Collection (serialized as described in 5.5.12).

See the MeterCollection Resource clause.

5.15.1.2 Operations

This Resource supports the Read, Update, and Delete operations. Create is supported through the `VolumeCollection` Resource.

In addition also the following custom operations are supported.

snapshot

/link@rel: `http://schemas.dmtf.org/cimi/2/action/snapshot`

This operation shall create a new `VolumeImage` from an existing `Volume`. This operation is defined within the `VolumeImage` Resource; see 5.15.7.1 for more details. Note that while this operation is performed against a `VolumeImage`, its presence in the `Volume` serialization is used to advertise support for the operation.

If the Volume is unable to accept operations at any point while it is creating the VolumeImage, the Volume shall be in the state "CAPTURING".

restore

/link@rel: `http://schemas.dmtf.org/cimi/2/action/restore`

This operation shall restore a `Volume` from a previously created `VolumeImage`.

Input parameters:

1) "image" - type: ref - mandatory
2) A reference to the Volume Image.

Output parameters: None.

During the processing of this operation, the `Volume` shall be in the "RESTORING" state.

Upon successful completion of this operation, the `Volume` shall again be in the state “AVAILABLE”.

HTTP protocol

To restore a `Volume`, a POST is sent to the "http://schemas.dmtf.org/cimi/2/action/restore" URI of the `Volume` where the HTTP request body shall be as described below.

JSON media type: application/json

JSON serialization:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/Action",
  "action": "http://schemas.dmtf.org/cimi/2/action/restore",
  "image": { "href" : string },
  "properties": { string: string, + } ?
  ...
}
```

XML media type: application/xml

XML serialization

```
<Action xmlns="http://schemas.dmtf.org/cimi/2">
  <action> http://schemas.dmtf.org/cimi/2/action/restore </action>
  <image href="xs:anyURI"/>
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties>
  <xs:any>*
</Action>
```

Where the "image" ref content is a reference to the `VolumeImage` to be used.

Upon successful processing of the request, the HTTP response body may be empty.

5.15.2 VolumeCollection Resource

A `VolumeCollection` Resource represents the Collection of `Volumes` within a Provider and follows the Collection pattern defined in clause 5.5.12.

5.15.2.1 Operations

NOTE The "add" operation requires that a VolumeTemplate be used (see 4.2.1.1).

5.15.3 VolumeTemplate Resource

This Resource captures the configuration values for realizing a Volume. A `VolumeTemplate` may be used to create multiple `Volumes`. Table 27 describes the `VolumeTemplate` attributes.

Table 27 – VolumeTemplate attributes

Name	VolumeTemplate	
Type URI	http://schemas.dmtf.org/cimi/2/VolumeTemplate	
Attribute	**Type**	**Description**
volumeConfig	*ref*	A reference to the `VolumeConfiguration` that is used to create a `Volume` from this `VolumeTemplate`. Note that the attributes of the `VolumeConfiguration` may be specified rather than a reference to an existing `VolumeConfiguration` Resource.
volumeImage	*ref*	A reference to the `VolumeImage` that is used to create a `Volume` from this `VolumeTemplate`.
meterTemplates	*Meter Templates[]*	A list of references to `MeterTemplates` that shall be used to create and connect a set of new `Meters` to the new `Volume`. Note that the attributes of the `MeterTemplate` may be specified rather than a reference to an existing `MeterTemplate` Resource.
eventLog Template	*ref*	A reference to an `EventLogTemplate` that shall be used to create and connect a new `EventLog` to the new `Volume`. Note that the attributes of the `EventLogTemplate` may be specified rather than a reference to an existing `EventLogTemplate` Resource.
genResourceMet adata	*ref*	A reference to a `ResourceMetadata` that shall be associated with every Volume generated from this Template.

When implementing or using `VolumeTemplate`, Providers and Consumers shall adhere to the syntax and semantics of its attributes as described in the above table as well as in the tables describing embedded Resources or related Collections.

5.15.3.1 Operations

This Resource supports the Read, Update, and Delete operations. Create is supported through the `VolumeTemplateCollection` Resource.

5.15.4 VolumeTemplateCollection Resource

A `VolumeTemplateCollection` Resource represents the Collection of `VolumeTemplate` Resources within a Provider and follows the Collection pattern defined in clause 5.5.12.

5.15.4.1 Operations

This Resource supports the Read and Update operations. Creation of new `VolumeTemplate` Resources is supported by the way of a POST to the "add" operation's URI as described in clause 4.2.1.1.

5.15.5 VolumeConfiguration Resource

The `VolumeConfiguration` Resource represents the set of configuration values needed to create a `Volume` with certain characteristics. `VolumeConfigurations` are created by Providers and may, at the Providers discretion, be created by Consumers.

Table 28 describes the `VolumeConfiguration` attributes.

Table 28 – VolumeConfiguration attributes

Name	VolumeConfiguration	
Type URI	http://schemas.dmtf.org/cimi/2/VolumeConfiguration	
Attribute	**Type**	**Description**
type	*URI*	A URI that indicates the type of `Volume` to be created. This specification defines the following URI: `http://schemas.dmtf.org/cimi/2/mapped`: Indicates a `Volume` that shall be used for shared storage that might be available to multiple `Machines`, but which does not require an explicit mount operation from within the guest operating system. Additional values may be defined. If certain types of `Volumes` require additional data, it is expected that this Resource is extended.
format	*string*	The format of the file system that is placed on `Volumes` created from this configuration. This attribute is only meaningful for `VolumeConfigurations` that describe block devices. This attribute is optional; the absence of this attribute indicates that Volumes created from this configuration are not formatted with a file system. Example values: "ext4," "ntfs."
capacity	*integer*	The default size in kilobytes, if limited, of the `Volume` created from this `VolumeConfiguration`.

5.15.5.1 Operations

This Resource supports the Read, Update, and Delete operations. Create is supported through the `VolumeConfigurationCollection` Resource.

5.15.6 VolumeConfigurationCollection Resource

A `VolumeConfigurationCollection` Resource represents the Collection of `VolumeConfiguration` Resources within a Provider and follows the Collection pattern defined in clause 5.5.12.

5.15.6.1 Operations

This Resource supports the Read and Update operations. Creation of new `VolumeImage` Resources is supported by the way of a POST to the "add" operations' URI as described in clause 4.2.1.1.

5.15.7 VolumeImage Resource

This Resource represents an image that could be placed on a preloaded volume. Table 29 describes the `VolumeImage` attributes.

Table 29 – VolumeImage attributes

Name	VolumeImage	
Type URI	http://schemas.dmtf.org/cimi/2/VolumeImage	
Attribute	**Type**	**Description**
state	*string*	The operational state of the `VolumeImage`. Allowed values are: **CREATING**: The `VolumeImage` is in the process of being created. **AVAILABLE**: The `VolumeImage` is available and ready for use. Unless otherwise specified, the `VolumeImage` shall initially be in this state after successful creation. **DELETING**: The `VolumeImage` is in the process of being deleted. **ERROR**: The Provider has detected an error in the `VolumeImage`. The operations that result in transitions to the above defined states are defined in clause 5.15.7.1
imageLocation	*URI*	A reference to the location of the binary data that makes up this image.

Name	VolumeImage	
Type URI	http://schemas.dmtf.org/cimi/2/VolumeImage	
Attribute	**Type**	**Description**
bootable	*boolean*	This property indicates whether Volumes created from this `VolumeImage` are bootable.

5.15.7.1 Operations

This Resource supports the Read, Update, and Delete operations. Create is supported through the `VolumeImageCollection` Resource.

5.15.8 VolumeImageCollection Resource

A `VolumeImageCollection` Resource represents the Collection of `VolumeImage` Resources within a Provider and follows the Collection pattern defined in clause 5.5.12.

5.15.8.1 Operations

This Resource supports the Read and Update operations. Creation of new `VolumeImage` Resources is supported by the way of a POST to the "add" operation's URI as described in clause 4.2.1.1.

During the creation of a new `VolumeImage` Resource, if the "imageLocation" attribute refers to an existing `Volume`, this operation shall be interpreted as a request to create a snapshot of the `Volume`. Once completed, the "imageLocation" attribute of the new `VolumeImage` Resource shall not refer to the original `Volume`; instead it shall refer to a static copy of the `Volume`. Additionally, the referenced Volume's `VolumeImageCollection` shall be updated to include a reference to this newly created snapshot `VolumeImage` Resource. During this process, the Provider may put the `Volume` into a "CAPTURING" state if necessary.

5.16 Network Resources and relationships

A Network is a logical construct that allows communication between defined Endpoints within a Segment. Each Segment uses a single, fixed, protocol to communicate and access is provided by associating an Endpoint with an Interface.

Only Endpoints within a Segment can communicate implicitly. All other communication must be explicitly enabled using Network Services.

- Each Network has one or more Segments
- Each Segment supports communication using a single protocol
- Each Segment may have one or more addressable Endpoints
- Each Endpoint is associated with a single Segment
- Each Endpoint may be associated with a single Interface
- An Interface can be associated with more than one Endpoint
- A Network may contain subordinate Networks to form hierarchical structures (similar to Systems)
- One or more Services may be associated with a Network to provide additional functionality

5.16.1 Network

Table 30 describes the `Network` Resource attributes.

Table 30 – Network attributes

Name	Network	
Type URI	http://schemas.dmtf.org/cimi/2/Network	
Attribute	**Type**	**Description**
state	*string*	The operational state of the `Network`. Allowed values are: **CREATING**: The `Network` is in the process of being created. **STARTING**: The `Network` is in the process of being started. **STARTED**: The `Network` is available and ready for use. **STOPPING**: The `Network` is in the process of being stopped. **STOPPED**: The `Network` is stopped and not available for use. **DELETING**: The `Network` is in the process of being deleted. **ERROR**: The Provider has detected an error in the `Network`. The operations that result in transitions to the above defined states are defined in clause 5.16.1.2. Clause 0 defines the initial state of a Network.
segments	*collection [Protocol Segment]*	A reference to a Collection of Segments contained within this Network.
services	*collection [Network Service]*	A reference to a Collection of Services that may be applied to this Network.
subnetworks	*collection [Network]*	A reference to a Collection of subordinate Networks contained within this Network.
meters	*collection [Meter]*	A reference to the list of `Meters` monitored for this `Network`.
eventLog	*ref*	A reference to the `EventLog` of this `Network`.

The Provider should supply at least one `Network` Resource in the CEP `Networks` Collection to represent communication channels that are external to the Consumers cloud. Typically this would be a connection to the Internet. As an alternative the Provider may supply a `NetworkTemplate` Resource by which such external Networks can be created when required.

When implementing or using `Network` Resources, Providers and Consumers shall adhere to the syntax and semantics of its attributes as described in Table 30 as well as in the tables describing embedded Resources or related Collections. 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.

5.16.1.1 Collections

The following clauses describe the Collection Resources that are components of `Networks`.

5.16.1.1.1 `segments` Collection

The Resource type for each item of this Collection is "`ProtocolSegment`". There is no accessory attribute for the items in this Collection, therefore it is a basic `ProtocolSegmentCollection`, as described in 5.16.6.

5.16.1.1.2 `services` Collection

The Resource type for each item of this Collection is "`NetworkService`". There is no accessory attribute for the items in this Collection, therefore it is a basic `NetworkServiceCollection`, as described in 5.16.18

5.16.1.1.3 `subnetworks` Collection

The Resource type for each item of this Collection is "`Network`". There is no accessory attribute for the items in this Collection, therefore it is a basic `NetworkCollection`, as described in 5.16.2.

5.16.1.1.4 `meters` Collection

The Resource type for each item of this Collection is "`Meter`" as defined in clause 5.17.3. There is no accessory attribute for the items in this Collection, therefore it is a basic `MeterCollection` as described in 5.5.12.

See the MeterCollection Resource clause.

5.16.1.2 Operations

`Network` Resources support the Read, Update, and Delete operations. Create is supported through the `NetworkCollection` Resource, as described in 5.16.2.

The following custom operations are also defined:

start

/link@rel: `http://schemas.dmtf.org/cimi/2/action/start`

This operation shall recursively start and enable all the components within a Network.

Input parameters: None.

Output parameters: None.

During the processing of this operation, the `Network` shall be in the "STARTING" state.

Upon successful completion of this operation, the `Network` shall be in the "STARTED" state.

HTTP protocol

To start a `Network`, a POST is sent to the "http://schemas.dmtf.org/cimi/2/action/start" URI of the `Network` where the HTTP request body shall be as described below.

JSON media type: application/json

JSON serialization:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/Action",
  "action": "http://schemas.dmtf.org/cimi/2/action/start",
  "properties": { string: string, + } ?
  ...
}
```

XML media type: application/xml

XML serialization

```
<Action xmlns="http://schemas.dmtf.org/cimi/2">
  <action> http://schemas.dmtf.org/cimi/2/action/start </action>
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties> ?
  <xs:any>*
</Action>
```

Upon successful processing of the request, the HTTP response body may be empty.

stop

/link@rel: `http://schemas.dmtf.org/cimi/2/action/stop`

This operation shall recursively stop and disable all components of a `Network`.

Input parameters: None.

Output parameters: None.

During the processing of this operation, the `Network` shall be in the "STOPPING" state.

Upon successful completion of this operation, the `Network` shall be in the "STOPPED" state.

HTTP protocol

To stop a `Network`, a POST is sent to the "http://schemas.dmtf.org/cimi/2/action/stop" URI of the `Network` where the HTTP request body shall be as described below.

JSON media type: application/json

JSON serialization:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/Action",
  "action": "http://schemas.dmtf.org/cimi/2/action/stop",
  "properties": { string: string, + } ?
  ...
}
```

XML media type: application/xml

XML serialization

```
<Action xmlns="http://schemas.dmtf.org/cimi/2">
  <action> http://schemas.dmtf.org/cimi/2/action/stop </action>
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties> ?
  <xs:any>*
</Action>
```

Upon successful processing of the request, the HTTP response body may be empty.

5.16.2 NetworkCollection Resource

A `NetworkCollection` Resource represents the Collection of `Networks` and follows the Collection pattern that is defined in clause 5.5.12. Operations

NOTE The "add" operation requires that a `NetworkTemplate` be used (see 5.16.3).

Upon successful processing of the "add" operation, unless otherwise specified by the way of the `NetworkTemplate` "initialState" attribute, the state of the new `Network` shall be the value of the DefaultInitialState capability of the `Network` Resource's `ResourceMetadata`, if defined. If no DefaultInitialState capability is defined, the default value shall be "STOPPED." The semantics of "initialState" shall be equivalent to the Provider issuing the appropriate actions against the new `Network` to move it into that state.

If a Provider is unable to change the state of the new `Network` to the appropriate "initialState" (either as specified by the `NetworkTemplate` or as implied by the previous stated rules), the `Network` creation shall fail.

5.16.3 NetworkTemplate Resource

The NetworkTemplate is a set of configuration values for realizing a `Network`. An instance of `NetworkTemplate` may be used to create multiple `Networks`. Table 31 describes the `NetworkTemplate` attributes.

Table 31 – NetworkTemplate attributes

Name	NetworkTemplate	
Type URI	http://schemas.dmtf.org/cimi/2/NetworkTemplate	
Attribute	**Type**	**Description**
initialState	*string*	Sets the initial state of a Network created using this Template. The allowed values are restricted to the non-transient states specified for the `state` attribute of the `Network` Resource, described in Table 30. Providers should advertise the list of available values via the `Network ResourceMetadata initialStates` Capability.
segments	*Protocol Segment[]*	A list of references to existing `ProtocolSegment` Resources to be inserted into the "`segments`" collection of the `Network` Resource created using this Template.
segmentTemplates	*Protocol Segment Template[]*	A list of references to `ProtocolSegmentTemplates`, from each of which a `ProtocolSegment` Resource is created and its reference inserted into the "`segments`" collection of the `Network` Resource created using this `NetworkTemplate`.
services	*Network Service[]*	A list of references to `NetworkService` Resources to be added to the "`services`" collection of the `Network` Resource created using this Template.
serviceTemplates	*Network Service Template[]*	A list of references to `NetworkServiceTemplates`, from each of which a `NetworkService` Resource is created and its reference inserted into the "`services`" collection of the `Network` Resource created using this Template.

Name	NetworkTemplate	
Type URI	http://schemas.dmtf.org/cimi/2/NetworkTemplate	
Attribute	**Type**	**Description**
subnetworks	*Network[]*	A list of references to `Network` Resources to be added to the `subnetworks` collection of the `Network` created from this `NetworkTemplate`
subnetworkTemplates	*Network Template[]*	A list of references to `NetworkTemplates`, from each of which a `Network` Resource is created and added to the `subnetworks` collection of the `Network` created using this `NetworkTemplate`.
meterTemplates	*Meter Template[]*	A list of references to `MeterTemplates` that shall be used to create and connect a set of new `Meters` to the new `Network`. Note that the attributes of the `MeterTemplate` may be specified rather than a reference to an existing `MeterTemplate` Resource.
eventLogTemplate	*ref*	A reference to an `EventLogTemplate` that shall be used to create and connect a new `EventLog` to the new `Network`. Note that the attributes of the `EventLogTemplate` may be specified rather than a reference to an existing `EventLogTemplate` Resource.

When implementing or using `NetworkTemplate`, Providers and Consumers shall adhere to the syntax and semantics of its attributes as described in Table 31 as well as in the tables describing embedded Resources or related CollectionsOperations

The `NetworkTemplate` Resource supports the Read, Update and Delete operations. Create is supported through the `NetworkTemplateCollection` Resource.

5.16.4 NetworkTemplateCollection Resource

A `NetworkTemplateCollection` Resource represents the Collection of `NetworkTemplates` within a Provider and follows the Collection pattern defined in clause 5.5.12.

5.16.4.1 Operations

The `NetworkTemplateCollection` Resource supports the Read and Update operations. Creation of new `NetworkTemplate` Resources is supported by the way of a POST to the "add" operation's URI as described in clause 4.2.1.1.

5.16.5 Segments

A Segment is an individual channel within a Network that utilizes a single communication protocol. Segments are `ProtocolSegment` Resources, the attributes of which are described in Table 32.

Table 32 – ProtocolSegment attributes

Name	ProtocolSegment	
Type URI	http://schemas.dmtf.org/cimi/2/ProtocolSegment	
Attribute	**Type**	**Description**
state	*string*	The operational state of the Segment. Allowed values are: **CREATING**: The Segment is in the process of being created. **STARTED**: The Segment is available (enabled) and ready for use. **STOPPED**: The Segment is stopped (disabled) and not available for use. **DELETING**: The Segment is in the process of being deleted. **ERROR**: The Provider has detected an error in the Segment. The operations that result in transitions to the above defined states are defined in clause 5.16.5.3. Clause 5.16.6.1 defines the initial state of a Segment.
protocol	*string*	The official name of the protocol supported by this segment. Allowed values are: **Ethernet**: As defined by IEEE 802.3. **IPv4**: Internet Protocol version 4, as defined in RFC 791. **IPv6**: Internet Protocol Version 6 as defined in RFC 2460.
noDefault Routing	*boolean*	If set to TRUE the default communication between Endpoints within the Segment is disabled. Communication between Endpoints in this case must be performed by a Service. The default value is FALSE which enables communication between endpoints.
endpoints	*collection [Protocol Endpoint]*	A reference to a list of references to Endpoints associated with this Segment.
parameters	*map*	A polymorphic attribute the contents of which depend on the specific network protocol. As examples this would include "netmask" for IPv4 and "bandwidth" for "Ethernet". See the adjacent tables for details of the data to be included
meters	*collection [Meter]*	A reference to the list of `Meters` monitored for this Segment.
eventLog	*ref*	A reference to the `EventLog` of this `Segment`.

5.16.5.1 When implementing or using `ProtocolSegment` Resources, Providers and Consumers shall adhere to the syntax and semantics of its attributes as described in Table 32 as well as in the tables describing embedded Resources or related Collections. **Protocol specific parameters**

Each Segment may require additional data that is specific to a communication protocol. This additional data is specified using the `parameters` attribute of the `ProtocolSegment`.This specification defines the following key – value pairs that must be supplied for the indicated protocols:

Table 33 - IPv6 ProtocolSegment parameters

Name	IPv6ProtocolParameters	
Key	**Value Type**	**Description**
prefixLength	*integer*	The length of the prefix for IPv6 addresses that is used to specify a subnet.
subnetAddress	*string*	The IPv6 subnet address for this subnet.

Table 34 – IPv4 ProtocolSegment parameters

Name	IPv4ProtocolParameters	
Key	**Value Type**	**Description**
netmask	*string*	The IPv4 subnetwork mask that defines the subnet.
subnetAddress	*string*	The IPv4 subnet address for this subnet.

Table 35 – Ethernet ProtocolSegment parameters

Name	EthernetProtocolParameters	
Key	**Value Type**	**Description**
speed	*integer*	The current bandwidth of the Segment in Bits per second. If no accurate determination of speed is possible this attribute should contain the nominal bandwidth.
mtu	*integer*	The active or negotiated maximum transmission unit (MTU) that can be supported by this Segment.

Note that Providers may support additional key – value pairs for the `parameter` attribute to extend the existing protocols. Consumers are not required to process any additional key – value pairs but must retrun them to the Provider in the serialization of `ProtocolSegments`.

5.16.5.2 Collections

The following clauses describe the Collection Resources that are components of `ProtocolSegments`.

5.16.5.2.1 `endpoints` Collection

The Resource type for each item of this Collection is a “`ProtocolEndpoint`” as defined in clause 5.16.9. There is no accessory attribute for the items in this Collection, therefore it is a basic `ProtocolEndpointCollection` Resource, serialized as described in 5.16.10.

5.16.5.2.2 `meters` Collection

The Resource type for each item of this Collection is “`Meter`” as defined in clause 5.17.3. There is no accessory attribute for the items in this Collection, therefore it is a basic `Meter` Collection (serialized as described in 5.5.12).

5.16.5.3 Operations

The `ProtocolSegment` Resource supports the Read, Update, and Delete operations. Create is supported through the `ProtocolSegmentCollection` Resource.

Deleting a `ProtocolSegment` shall remove that Segment from the global (Cloud Entry Point) `ProtocolSegmentCollection` and also all references to the Segment in Collections of other Resources (e.g.from corresponding Network `segments` Collection).

The following custom operations are also defined:

start

/link@rel: `http://schemas.dmtf.org/cimi/2/action/start`

This operation shall start a `ProtocolSegment`.

Input parameters: None.

Output parameters: None.

Upon successful completion of this operation, the `ProtocolSegment` shall be in the "STARTED" state.

HTTP protocol

To start a `ProtocolSegment`, a POST is sent to the "http://schemas.dmtf.org/cimi/2/action/start" URI of the `ProtocolSegment` where the HTTP request body shall be as described below.

JSON media type: application/json

JSON serialization:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/Action",
  "action": "http://schemas.dmtf.org/cimi/2/action/start",
  "properties": { string: string, + } ?
  ...
}
```

XML media type: application/xml

XML serialization

```
<Action xmlns="http://schemas.dmtf.org/cimi/2">
  <action> http://schemas.dmtf.org/cimi/2/action/start </action>
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties> ?
  <xs:any>*
</Action>
```

Upon successful processing of the request, the HTTP response body may be empty.

stop

/link@rel: `http://schemas.dmtf.org/cimi/2/action/stop`

This operation shall stop a `ProtocolSegment`.

Input parameters: None.

Output parameters: None.

Upon successful completion of this operation, the `ProtocolSegment` shall be in the "STOPPED" state.

HTTP protocol

To stop a `ProtocolSegment`, a POST is sent to the "http://schemas.dmtf.org/cimi/2/action/stop" URI of the `ProtocolSegment` where the HTTP request body shall be as described below.

JSON media type: application/json

JSON serialization:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/Action",
  "action": "http://schemas.dmtf.org/cimi/2/action/stop",
  "properties": { string: string, + } ?
  ...
}
```

XML media type: application/xml

XML serialization

```
<Action xmlns="http://schemas.dmtf.org/cimi/2">
  <action> http://schemas.dmtf.org/cimi/2/action/stop </action>
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties> ?
  <xs:any>*
</Action>
```

Upon successful processing of the request, the HTTP response body may be empty.

5.16.6 ProtocolSegmentCollection Resource

A `ProtocolSegmentCollection` Resource represents the Collection of `ProtocolSegments` within a Provider and follows the Collection pattern defined in clause 5.5.12.

5.16.6.1 Operations

NOTE The "add" operation requires that a ProtocolSegmentTemplate be used (see clause 5.16.7).

Upon successful processing of the "add" operation, unless otherwise specified by the `ProtocolSegmentTemplate` "initialState" attribute, the state of the new `ProtocolSegment` shall be the value of the DefaultInitialState capability of the `ProtocolSegment` Resource's `ResourceMetadata`, if defined. If no DefaultInitialState capability is defined, the default value shall be "STOPPED." The semantics of "initialState" shall be equivalent to the Provider issuing the appropriate actions against the new `ProtocolSegment` to move it into that state.

If a Provider is unable to change the state of the new `ProtocolSegment` to the appropriate "initialState" (either as specified by the `ProtocolSegmentTemplate` or as implied by the previous stated rules), the `ProtocolSegment` creation shall fail.

5.16.7 ProtocolSegmentTemplate Resource

The `ProtocolSegmentTemplate` is a set of configuration values for realizing a `ProtocolSegment`. A `ProtocolSegmentTemplate` may be used to create multiple `ProtocolSegments`. Table 36 describes the `ProtocolSegmentTemplate` attributes.

3346 **Table 36 – ProtocolSegmentTemplate attributes**

Name	ProtocolSegmentTemplate	
Type URI	http://schemas.dmtf.org/cimi/2/ProtocolSegmentTemplate	
Attribute	**Type**	**Description**
network	*ref*	A reference to the Network to which the Segment created using this Template belongs. If this Template is used to create a new Segment through the global (Cloud Entry Point) `ProtocolSegmentCollection`, this attribute shall be present. If this Template is referenced from a `NetworkTemplate` and used to create a new Segment during the creation of a Network, this attribute shall either be absent or have the same value as the "`id`" attribute of the `Network` to which this Segment is being added.
initialState	*string*	Sets the initial state of the Segment created using this Template. The allowed values are restricted to the non-transient states specified for the `state` attribute of the `ProtocolSegment` Resource, described in 5.16.5. Providers should advertise the list of available values via the `ProtocolSegment` `ResourceMetadata` `initialStates` Capability.
protocol	*string*	Sets the protocol supported by the Segment created using this Template. The allowed values are those specified for the `protocol` attribute of the `ProtocolSegment` Resource, described in clause 5.16.5.
noDefault Routing	*boolean*	Enables or disables default routing for the Segment created using this Template. Values are as described for the `noDefaultRouting` attribute of the `ProtocolSegment` Resource, described in clause 5.16.5.
endpoints	*Protocol Endpoint[]*	A list of references to `ProtocolEndpoints` to be inserted into the `endpoints` Collection of the Segment created using this Template.
endpoint Templates	*Protocol Endpoint Template[]*	A list of references to `ProtocolEndpointTemplates` that specify a set of Endpoints to be created and inserted into the `endpoints` Collection for the Segement created using this Template. Note that the Template attributes may be explicitly listed rather than providing a reference to an existing `ProtocolEndpointTemplate` Resource.
parameters	*map*	A polymorphic attribute the contents of which depend on the specific protocol supported. The allowed key – value pairs are as specified in section 5.16.5.1.
meterTemplates	*meterTemplates []*	A list of references to `MeterTemplates` that shall be used to create and connect a set of new `Meters` to the new `ProtocolSegment`. Note that the attributes of the `MeterTemplate` may be specified rather than a reference to an existing `MeterTemplate` Resource.
eventLogTemplate	*ref*	A reference to an `EventLogTemplate` that shall be used to create and connect a new `EventLog` to the new `ProtocolSegment`. Note that the attributes of the `EventLogTemplate` may be specified rather than a reference to an existing `EventLogTemplate` Resource.

When implementing or using ProtocolSegmentTemplate Resources, Providers and Consumers shall adhere to the syntax and semantics of its attributes as described in Table 36 as well as in the tables describing embedded Resources or related Collections.

5.16.7.1 Collections

The ProtocolSegmentTemplate.Resource has no attributes of type Collection.

5.16.7.2 Operations

The ProtocolSegmentTemplate Resource supports the Read, Update, and Delete operations. Create is supported through the ProtocolSegmentTemplateCollection Resource.

5.16.8 ProtocolSegmentTemplateCollection Resource

A ProtocolSegmentTemplateCollection Resource represents the Collection of ProtocolSegmentTemplates within a Provider and follows the Collection pattern defined in clause 5.5.12.

5.16.8.1 Operations

The ProtocolSegmentTemplateCollection Resource supports the Read and Update operations. Creation of new ProtocolSegmentTemplate Resources is supported by the way of a POST to the "add" operation's URI as described in clause 4.2.1.1.

5.16.9 Endpoints

An Endpoint is an addressable element within a protocol that is a source, destination , or source and destination for communication. Endpoints are ProtocolEndpoint Resources, the attributes of which are described in Table 37.

Table 37 – ProtocolEndpoint attributes

Name	ProtocolSegment	
Type URI	http://schemas.dmtf.org/cimi/2/ProtocolEndpoint	
Attribute	**Type**	**Description**
state	*string*	The operational state of the Endpoint. Allowable values are: **CREATING**: The Endpoint is in the process of being created. **ENABLED**: The Endpoint is available and ready for use. **DISABLED**: The Endpoint is not available for use. **DELETING**: The Endpoint is in the process of being deleted. **ERROR**: The Provider has detected an error in the Endpoint. The operations that result in transitions to the above defined states are defined in clause 5.16.9.3. Clause 5.16.10.1 defines the initial state of an Endpoint.
protocol	*string*	The official name of the protocol supported by this segment. This attribute is intended as a convienience only and if specified its value must be identical to the value of the protocol attribute of the Segment with which the Endpoint is associated. Possible values are those specified in the ProtocolSegment Resource described in section 5.16.5.
address	*string*	The address assigned to this Endpoint in the format required by the supported protocol.

origin	*string*	A string representing how protocol specific data is assigned to this Endpoint. Allowable values are: [**STATIC** | **DYNAMIC**] In general the Consumer is responsible for assignment of static data, usually from within the guest software. The Provider may assign data dynamically when the end point is created, or it may be assigned via a Service associated with the Segment to which the Endpoint belongs. (E.g. DHCP).
interface	*Network Interface*	A reference to the Interface that is used to connect to the Network using this Endpoint.
parameters	*map*	A polymorphic attribute the contents of which depend on the specific network protocol. As examples this would include "netmask" for IPv4 and "bandwidth" for "Ethernet". See the adjacent tables for details of the data to be included
meters	*collection [Meter]*	A reference to the list of `Meters` monitored for this Endpoint.
eventLog	*ref*	A reference to the `EventLog` of this `Endpoint`.

When implementing or using `ProtocolEndpoint`, Providers and Consumers shall adhere to the syntax and semantics of its attributes as described in Table 37 as well as in the tables describing embedded Resources or related Collections.

5.16.9.1 Protocol specific parameters

Each Endpoint may require additional data that is specific to the communication protocol supported. This additional data is specified using the `parameters` attribute of a `ProtocolEndpoint`.This specification defines the following key – value pairs that provide supplemental information for Endpoints of specific protocol types:

Table 38 - IPv6 ProtocolEndpoint parameters

Name	IPv6ProtocolEndpointParameters	
Key	**Value Type**	**Description**
addressType	*string*	The IPv6 address type as specified by RFC4291, Section 2.4. Allowed values: [Unspecified | Loopback | Multicast | Link Local Unicast | Global Unicast | Embedded IPv4 Address | Site Local Unicast] If specified this value must match the type of address specified by the `address` attribute of the IPv6 Endpoint with which it is associated.
prefixLength	*integer*	The length of the prefix for IPv6 addresses that is used to specify a subnet.

Table 39 – IPv4 ProtocolEndpoint parameters

Name	IPv4ProtocolEndpointParameters	
Key	**Value Type**	**Description**
hostname	*string*	The DNS resolvable name associated with this address.

Table 40 – Ethernet ProtocolEndpoint parameters

Name	EthernetProtocolEndpointParameters	
Key	**Value Type**	**Description**
aliases	*string[]*	Other unicast addresses that may be used to communicate with the Endpoint
groupAddresses	*string[]*	Multicast addresses to which the Endpoint listens.

Note that Providers may support additional key – value pairs for the `parameter` attribute to extend the existing protocols. Consumers are not required to process any additional key – value pairs but must retrun them to the Provider in the serialization of `ProtocolEndpoints`.

5.16.9.2 Collections

The following clauses describe the Collection Resources that are components of `ProtocolEndpoints`.

5.16.9.2.1 `meters` Collection

The Resource type for each item of this Collection is "`Meter`" as defined in clause 5.17.3. There is no accessory attribute for the items in this Collection, therefore it is a basic `Meter` Collection (serialized as described in 5.5.12).

5.16.9.3 Operations

The `ProtocolEndpoints` Resource supports the Read, Update, and Delete operations. Create is supported through the `ProtocolEndpointCollection` Resource.

Deleting a `ProtocolEndpoint` shall remove that Endpoint from the global (Cloud Entry Point) `ProtocolEndpointCollection`. Additionally, references to the Endpoint in `ProtocolEndpointCollections` of all other Resources (e.g. `ProtocolSegments`, `NetworkServices`) must be removed.

The following custom operations are also defined:

enable

/link@rel: `http://schemas.dmtf.org/cimi/2/action/enable`

This operation shall enable a `ProtocolEndpoint`.

Input parameters: None.

Output parameters: None.

Upon successful completion of this operation, the `ProtocolEndpoint` shall be in the "ENABLED" state.

HTTP protocol

To enable a `ProtocolEndpoint`, a POST is sent to the "http://schemas.dmtf.org/cimi/2/action/enable" URI of the `ProtocolEndpoint` where the HTTP request body shall be as described below.

JSON media type: application/json

JSON serialization:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/Action",
  "action": "http://schemas.dmtf.org/cimi/2/action/enable",
  "properties": { string: string, + } ?
  ...
}
```

XML media type: application/xml

XML serialization

```
<Action xmlns="http://schemas.dmtf.org/cimi/2">
  <action> http://schemas.dmtf.org/cimi/2/action/enable </action>
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties> ?
  <xs:any>*
</Action>
```

Upon successful processing of the request, the HTTP response body may be empty.

disable

/link@rel: `http://schemas.dmtf.org/cimi/2/action/disable`

This operation shall disable a `ProtocolEndpoint`.

Input parameters: None.

Output parameters: None.

Upon successful completion of this operation, the `ProtocolEndpoint` shall be in the "DISABLED" state.

HTTP protocol

To stop a `ProtocolEndpoint`, a POST is sent to the "http://schemas.dmtf.org/cimi/2/action/disable" URI of the `ProtocolEndpoint` where the HTTP request body shall be as described below.

JSON media type: application/json

JSON serialization:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/Action",
  "action": "http://schemas.dmtf.org/cimi/2/action/disable",
  "properties": { string: string, + } ?
  ...
}
```

XML media type: application/xml

XML serialization

```
<Action xmlns="http://schemas.dmtf.org/cimi/2">
  <action> http://schemas.dmtf.org/cimi/2/action/disable </action>
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties> ?
  <xs:any>*
</Action>
```

Upon successful processing of the request, the HTTP response body may be empty.

5.16.10 ProtocolEndpointCollection Resource

A `ProtocolEndpointCollection` Resource represents the Collection of `ProtocolEndpoints` within a Provider and follows the Collection pattern defined in clause 5.5.12.

5.16.10.1 Operations

NOTE The "add" operation requires that a ProtocolEndpointTemplate be used (see clause 5.16.11).

Upon successful processing of the "add" operation, unless otherwise specified by the `ProtocolEndpointTemplate` "initialState" attribute, the state of the new `ProtocolEndpoint` shall be the value of the DefaultInitialState capability of the `ProtocolEndpoint` Resource's `ResourceMetadata`, if defined. If no DefaultInitialState capability is defined, the default value shall be "DISABLED." The semantics of "initialState" shall be equivalent to the Provider issuing the appropriate actions against the new `ProtocolEndpoint` to move it into that state.

If a Provider is unable to change the state of the new `ProtocolEndpoint` to the appropriate "initialState" (either as specified by the `ProtocolEndpointTemplate` or as implied by the previous stated rules), the `ProtocolEndpoint` creation shall fail.

5.16.11 ProtocolEndpointTemplate Resource

The `ProtocolEndpointTemplate` is a set of configuration values for realizing a `ProtocolEndpoint`. A `ProtocolEndpointTemplate` may be used to create multiple `ProtocolEndpoints`. Table 41 describes the `ProtocolEndpointTemplate` attributes.

Table 41 – ProtocolEndpointTemplate attributes

Name	ProtocolEndpointTemplate	
Type URI	http://schemas.dmtf.org/cimi/2/ProtocolEndpointTemplate	
Attribute	**Type**	**Description**
initialState	*string*	Sets the initial state of the Endpoint created using this Template. The allowed values are restricted to the non-transient states specified for the `state` attribute of the `ProtocolEndpoint` Resource, described in clause 5.16.9. Providers should advertise the list of available values via the `ProtocolEndpoint ResourceMetadata initialStates` Capability.

Name	ProtocolEndpointTemplate	
Type URI	http://schemas.dmtf.org/cimi/2/ProtocolEndpointTemplate	
Attribute	**Type**	**Description**
address	*string*	If the `origin` attribute value is "STATIC" this attribute contains the address to be assigned to this Endpoint in the format required by the supported protocol. If the `origin` attribute value is "DYNAMIC" this attribute must not be supplied by the Template.
origin	*string*	A string representing how protocol specific data is assigned to this Endpoint. Allowable values are: [**STATIC** | **DYNAMIC**] If the value of this attribute is "STATIC" then all protocol specific data for thei Endpoint must be supplied by this Template. If the value of this attribute is "DYNAMIC" then the protocol specific data for this Endpoint is allocated by other mechanisms and must not be supplied by this Template.
interface	*Network Interface*	A reference to a `NetworkInterface` Resource with which this new Endpoint is associated.
parameters	*map*	A polymorphic attribute the contents of which depend on the specific protocol supported. The allowed key – value pairs are as specified in clause 5.16.9. Whether this data is required to be supplied by this Template is determined by the value of the "`origin`" attribute described above.
meterTemplates	*MeterTemplate[]*	A list of references to `MeterTemplates` that shall be used to create and connect a set of new `Meters` to the new `ProtocolEndpoint`. Note that the attributes of the `MeterTemplate` may be specified rather than a reference to an existing `MeterTemplate` Resource.
eventLogTemplate	*ref*	A reference to an `EventLogTemplate` that shall be used to create and connect a new `EventLog` to the new `ProtocolEndpoint`. Note that the attributes of the `EventLogTemplate` may be specified rather than a reference to an existing `EventLogTemplate` Resource.

When implementing or using `ProtocolEndpointTemplate` Resources, Providers and Consumers shall adhere to the syntax and semantics of its attributes as described in Table 41 as well as in the tables describing embedded Resources or related Collections.

5.16.11.1 Collections

The `ProtocolEndpointTemplate` Resource has no attributes of type Collection.

5.16.11.2 Operations

The `ProtocolEndpointTemplate` Resource supports the Read, Update, and Delete operations. Create is supported through the `ProtocolEndpointTemplateCollection` Resource.

5.16.12 ProtocolEndpointTemplateCollection Resource

A `ProtocolEndpointTemplateCollection` Resource represents the Collection of `ProtocolEndpointTemplates` within a Provider and follows the Collection pattern defined in clause 5.5.12.

5.16.12.1 Operations

The `ProtocolEndpointTemplateCollection` Resource supports the Read and Update operations. Creation of new `ProtocolEndpointTemplate` Resources is supported by the way of a POST to the "add" operation's URI as described in clause 4.2.1.1.

5.16.13 Interfaces

An Interface provides a connection to a Network by associating Endpoints with Machines.The model is basically that of a virtual Network Interface Card (vNIC) that can support multiple communication protocols at multiple levels. Interfaces are `NetworkInterface` Resources, the attributes of which are described in Table 42.

Table 42 – NetworkInterface attributes

Name	NetworkInterface	
Type URI	http://schemas.dmtf.org/cimi/2/NetworkInterface	
Attribute	**Type**	**Description**
state	*string*	The operational state of the Interface. Allowable values are: **CREATING**: The Interface is in the process of being created. **ENABLED**: The Interface is available and ready for use. **DISABLED**: The Interface is not available for use. **DELETING**: The Interface is in the process of being deleted. **ERROR**: The Provider has detected an error in the Interface. The operations that result in transitions to the above defined states are defined in clause 5.16.13.2. Clause 5.16.14.1 defines the initial state of a Interface.
endpoints	*collection [Protocol Endpoint]*	A reference to a list of references to `ProtocolEndpoints` this Interface supports. Note: This Collection represents an association between the Interface and a list of Endpoints in one or more Segments.
speed	*integer*	The current bandwidth of the Interface in Bits per Second. For Interfaces that vary in bandwidth or for those where no accurate estimation can be made, this attribute should contain the nominal bandwidth..
mtu	*integer*	The size in bytes of the active or negotiated maximum transmission unit (MTU) that can be supported by this Interface.
meters	*collection [Meter]*	A reference to the list of `Meters` monitored for this Interface.
eventLog	*ref*	A reference to the `EventLog` of this Interface.

When implementing or using `NetworkInterface`, Providers and Consumers shall adhere to the syntax and semantics of its attributes as described in Table 42 as well as in the tables describing embedded Resources or related Collections.

5.16.13.1 Collections

The following clauses describe the Collection Resources that are components of `NetworkInterfaces`.

5.16.13.1.1 **`meters`** Collection

The Resource type for each item of this Collection is “`Meter`” as defined in clause 5.17.3. There is no accessory attribute for the items in this Collection, therefore it is a basic `Meter` Collection (serialized as described in 5.5.12).

5.16.13.2 Operations

The `NetworkInterfaces` Resource supports the Read, Update, and Delete operations. Create is supported through the `NetworkInterfaceCollection` Resource.

Deleting a `NetworkInterface` shall remove that Endpoint from the global (Cloud Entry Point) `NetworkInterfaceCollection`. Additionally, references to the Endpoint in `NetworkInterfaceCollections` of all other Resources (e.g. `ProtocolEndpoints`, `NetworkServices`) must be removed.

The following custom operations are also defined:

enable

/link@rel: `http://schemas.dmtf.org/cimi/2/action/enable`

This operation shall enable a `NetworkInterface`.

Input parameters: None.

Output parameters: None.

Upon successful completion of this operation, the `NetworkInterface` shall be in the "ENABLED" state.

HTTP protocol

To enable a `NetworkInterface`, a POST is sent to the "http://schemas.dmtf.org/cimi/2/action/enable" URI of the `NetworkInterface` where the HTTP request body shall be as described below.

JSON media type: application/json

JSON serialization:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/Action",
  "action": "http://schemas.dmtf.org/cimi/2/action/enable",
  "properties": { string: string, + } ?
  ...
}
```

XML media type: application/xml

XML serialization

```
<Action xmlns="http://schemas.dmtf.org/cimi/2">
  <action> http://schemas.dmtf.org/cimi/2/action/enable </action>
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties> ?
  <xs:any>*
</Action>
```

Upon successful processing of the request, the HTTP response body may be empty.

disable

/link@rel: `http://schemas.dmtf.org/cimi/2/action/disable`

This operation shall disable a `NetworkInterface`.

Input parameters: None.

Output parameters: None.

Upon successful completion of this operation, the `NetworkInterface` shall be in the "DISABLED" state.

HTTP protocol

To stop a `NetworkInterface`, a POST is sent to the "http://schemas.dmtf.org/cimi/2/action/disable" URI of the `NetworkInterface` where the HTTP request body shall be as described below.

JSON media type: application/json

JSON serialization:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/Action",
  "action": "http://schemas.dmtf.org/cimi/2/action/disable",
  "properties": { string: string, + } ?
  ...
}
```

XML media type: application/xml

XML serialization

```
<Action xmlns="http://schemas.dmtf.org/cimi/2">
  <action> http://schemas.dmtf.org/cimi/2/action/disable </action>
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties> ?
  <xs:any>*
</Action>
```

Upon successful processing of the request, the HTTP response body may be empty.

5.16.14 NetworkInterfaceCollection Resource

A `NetworkInterfaceCollection` Resource represents the Collection of `NetworkInterfaces` within a Provider and follows the Collection pattern defined in clause 5.5.12

5.16.14.1 Operations

NOTE The "add" operation requires that a NetworkInterfaceTemplate be used (see clause 5.16.15).

Upon successful processing of the "add" operation, unless otherwise specified by the `NetworkInterfaceTemplate` "initialState" attribute, the state of the new `NetworkInterface` shall be the value of the DefaultInitialState capability of the `NetworkInterface` Resource's `ResourceMetadata`, if defined. If no DefaultInitialState capability is defined, the default value shall be "DISABLED." The semantics of "initialState" shall be equivalent to the Provider issuing the appropriate actions against the new `NetworkInterface` to move it into that state.

If a Provider is unable to change the state of the new `NetworkInterface` to the appropriate "initialState" (either as specified by the `NetworkInterfaceTemplate` or as implied by the previous stated rules), the `NetworkInterface` creation shall fail.

5.16.15 NetworkInterfaceTemplate Resource

The `NetworkInterfaceTemplate` is a set of configuration values for realizing a `NetworkInterface`. A `NetworkInterfaceTemplate` may be used to create multiple `NetworkInterfaces`. Table 43 describes the `NetworkInterfaceTemplate` attributes.

Table 43 – NetworkInterfaceTemplate attributes

Name	NetworkInterfaceTemplate	
Type URI	http://schemas.dmtf.org/cimi/2/NetworkInterfaceTemplate	
Attribute	**Type**	**Description**
initialState	*string*	Sets the initial state of the Endpoint created using this Template. The allowed values are restricted to the non-transient states specified for the `state` attribute of the `NetworkInterface` Resource, described in 5.16.13 . Providers should advertise the list of available values via the `NetworkInterface` `ResourceMetadata` `initialStates` Capability.
endpoints	*collection [Protocol Endpoint]*	A reference to a list of references to `ProtocolEndpoints` this Interface supports. Note: This Collection represents an association between the Interface and a list of Endpoints in one or more Segments.
speed	*integer*	The initial bandwidth of the Interface in Bits per Second.
mtu	*integer*	The size in bytes of the initial maximum transmission unit (MTU) that can be supported by this Interface.
meterTemplates	*meterTemplates []*	A list of references to `MeterTemplates` that shall be used to create and connect a set of new `Meters` to the new `NetworkInterface`. Note that the attributes of the `MeterTemplate` may be specified rather than a reference to an existing `MeterTemplate` Resource.
eventLogTemplate	*ref*	A reference to an `EventLogTemplate` that shall be used to create and connect a new `EventLog` to the new `NetworkInterface`. Note that the attributes of the `EventLogTemplate` may be specified rather than a reference to an existing `EventLogTemplate` Resource.

When implementing or using `NetworkInterfaceTemplate` Resources, Providers and Consumers shall adhere to the syntax and semantics of its attributes as described in **Table 43** as well as in the tables describing embedded Resources or related Collections.

5.16.15.1 Collections

The following clauses describe Collection Resources that are components of `NetworkInterfaceTemplates`.

5.16.15.1.1 `endpoints` Collection

The Resource type for each item of this Collection is "`ProtocolEndpoint`" as defined in clause 5.16.9. There is no accessory attribute for the items in this Collection, therefore it is a basic `ProtocolEndpointCollection` (serialized as described in 5.16.10).

5.16.15.2 Operations

The `NetworkInterfaceTemplate` Resource supports the Read, Update, and Delete operations. Create is supported through the `NetworkInterfaceTemplateCollection` Resource.

5.16.16 NetworkInterfaceTemplateCollection Resource

A `NetworkInterfaceTemplateCollection` Resource represents the Collection of `NetworkInterfaceTemplates` within a Provider and follows the Collection pattern defined in clause 5.5.12.

5.16.16.1 Operations

The `NetworkInterfaceTemplateCollection` Resource supports the Read and Update operations. Creation of new `NetworkInterfaceTemplate` Resources is supported by the way of a POST to the "add" operation's URI as described in clause 4.2.1.1.

5.16.17 Services

Services provide all additional functionality within Networks beyond basic rounting within a single Segment. Services can be applied to individual Segments or Endpoints, collections of Segments or Endpoints, or combinations of these elements. The actual function provide by a Service is determined by policies (see clause 5.16.21). Services are `NetworkService` Resources, the attributes of which are described in Table 44.

Table 44 – NetworkService attributes

Name	NetworkService	
Type URI	http://schemas.dmtf.org/cimi/2/NetworkService	
Attribute	**Type**	**Description**
state	*string*	The operational state of the Service. Allowed values are: **CREATING**: The Service is in the process of being created. **STARTED**: The Service is available (enabled) and ready for use. **STOPPED**: The Service is stopped (disabled) and not available for use. **DELETING**: The Service is in the process of being deleted. **ERROR**: The Provider has detected an error in the Service. The operations that result in transitions to the above defined states are defined in clause 5.17. Clause 5.16.18.1 defines the initial state of a Service.
type	*string*	Indicates the type of service provided by this `NetworkService`. Allowed values: [Load Balancer | QoS | Firewall | VPN | DHCP | DNS | NAT | Gateway | Layer4 Port Forwarding | IP Routing | Virtual Network Device | Other]
endpoints	*collection [Protocol Endpoint]*	A reference to a list of references to individual Endpoints to which the Service is provided.
segments	*collection [Protocol Segment]*	A reference to a list of references to complete Segments to which the service is provided. The Service is provided to all Endpoints within each Segment.
policies	*map*	*** TBD *** Format & requirements yet to be determined form NSMWG work
meters	*collection [Meter]*	A reference to the list of `Meters` monitored for this Service.
eventLog	*ref*	A reference to the `EventLog` of this `Service`.

When implementing or using `NetworkService` Resources, Providers and Consumers shall adhere to the syntax and semantics of its attributes as described in **Table 44** as well as in the tables describing embedded Resources or related Collections.

5.16.17.1 Collections

The following clauses describe the Collection Resources that are components of `NetworkServices`.

5.16.17.1.1 `endpoints` Collection

The Resource type for each item of this Collection is a “`ProtocolEndpoint`” as defined in clause 5.16.9. There is no accessory attribute for the items in this Collection, therefore it is a basic `ProtocolEndpointCollection` Resource, serialized as described in 5.16.10.

5.16.17.1.2 `segments` Collection

The Resource type for each item of this Collection is a “`ProtocolSegment`” as defined in clause 5.16.55.16.9. There is no accessory attribute for the items in this Collection, therefore it is a basic `ProtocolSegmentCollection` Resource, serialized as described in 5.16.6.

5.16.17.1.3 `meters` Collection

The Resource type for each item of this Collection is “`Meter`” as defined in clause 5.17.3. There is no accessory attribute for the items in this Collection, therefore it is a basic `Meter` Collection (serialized as described in 5.5.12).

5.16.17.2 Operations

The `NetworkService` Resource supports the Read, Update, and Delete operations. Create is supported through the `NetworkServiceCollection` Resource.

Deleting a `NetworkService` shall remove that Service from the global (Cloud Entry Point) `NetworkServiceCollection` and also all references to the Service in Collections of other Resources (e.g.from corresponding Network `services` Collections).

The following custom operations are also defined:

start

/link@rel: `http://schemas.dmtf.org/cimi/2/action/start`

This operation shall start a `NetworkService`.

Input parameters: None.

Output parameters: None.

Upon successful completion of this operation, the `NetworkService` shall be in the "STARTED" state.

HTTP protocol

To start a `NetworkService`, a POST is sent to the "http://schemas.dmtf.org/cimi/2/action/start" URI of the `NetworkService` where the HTTP request body shall be as described below.

JSON media type: application/json

JSON serialization:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/Action",
  "action": "http://schemas.dmtf.org/cimi/2/action/start",
  "properties": { string: string, + } ?
  ...
}
```

XML media type: application/xml

XML serialization

```
<Action xmlns="http://schemas.dmtf.org/cimi/2">
  <action> http://schemas.dmtf.org/cimi/2/action/start </action>
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties> ?
  <xs:any>*
</Action>
```

Upon successful processing of the request, the HTTP response body may be empty.

stop

/link@rel: `http://schemas.dmtf.org/cimi/2/action/stop`

This operation shall stop a `NetworkService`.

Input parameters: None.

Output parameters: None.

Upon successful completion of this operation, the `NetworkService` shall be in the "STOPPED" state.

HTTP protocol

To stop a `NetworkService`, a POST is sent to the "http://schemas.dmtf.org/cimi/2/action/stop" URI of the `NetworkService` where the HTTP request body shall be as described below.

JSON media type: application/json

JSON serialization:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/Action",
  "action": "http://schemas.dmtf.org/cimi/2/action/stop",
  "properties": { string: string, + } ?
  ...
}
```

XML media type: application/xml

XML serialization

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

```
  <action> http://schemas.dmtf.org/cimi/2/action/stop </action>
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties> ?
  <xs:any>*
</Action>
```

Upon successful processing of the request, the HTTP response body may be empty.

5.16.18 NetworkServiceCollection Resource

A `NetworkServiceCollection` Resource represents the Collection of `NetworkServices` within a Provider and follows the Collection pattern defined in clause 5.5.12. This Resource shall be serialized as follows:

JSON serialization:

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

XML serialization:

```
<Collection
resourceURI="http://schemas.dmtf.org/cimi/2/NetworkServiceCollection"
    xmlns="http://schemas.dmtf.org/cimi/2">
  <id> xs:anyURI </id>
  <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>
```

5.16.18.1 Operations

NOTE The "add" operation requires that a NetworkServiceTemplate be used (see clause 5.16.19).

Upon successful processing of the "add" operation, unless otherwise specified by the `NetworkServiceTemplate` "initialState" attribute, the state of the new `NetworkService` shall be the value of the DefaultInitialState capability of the `NetworkService` Resource's `ResourceMetadata`, if defined. If no DefaultInitialState capability is defined, the default value shall be "STOPPED." The semantics of "initialState” shall be equivalent to the Provider issuing the appropriate actions against the new `NetworkService` to move it into that state.

If a Provider is unable to change the state of the new `NetworkService` to the appropriate "initialState" (either as specified by the `NetworkServiceTemplate` or as implied by the previous stated rules), the `NetworkService` creation shall fail.

5.16.19 NetworkServiceTemplate Resource

The `NetworkServiceTemplate` is a set of configuration values for realizing a `NetworkService`. A `NetworkServiceTemplate` may be used to create multiple `NetworkServices`. Table 45 describes the `NetworkServiceTemplate` attributes.

Table 45 – NetworkServiceTemplate attributes

Name	NetworkServiceTemplate	
Type URI	http://schemas.dmtf.org/cimi/2/NetworkServiceTemplate	
Attribute	**Type**	**Description**
network	*ref*	A reference to the Network to which the Service created using this Template belongs. If this Template is used to create a new Service through the global (Cloud Entry Point) `NetworkServiceCollection`, this attribute shall be present. If this Template is referenced from a `NetworkTemplate` and used to create a new Service during the creation of a Network, this attribute shall either be absent or have the same value as the "`id`" attribute of the `Network` to which this Service is being added.
initialState	*string*	Sets the initial state of the Service created using this Template. The allowed values are restricted to the non-transient states specified for the `state` attribute of the `NetworkService` Resource, described in clause 5.16.17. Providers should advertise the list of available values via the `NetworkService ResourceMetadata initialStates` Capability.
type	*string*	Sets the protocol supported by the Service created using this Template. The allowed values are those specified for the `protocol` attribute of the `NetworkService` Resource, described in 5.16.17
endpoints	*Protocol Endpoint[]*	A list of references to `ProtocolEndpoints` to be inserted into the `endpoints` Collection of the Service created using this Template.
segments	*Protocol Segment[]*	A list of references to `ProtocolSegments` to be inserted into the `segments` Collection of the Service created using this Template.

Name	NetworkServiceTemplate	
Type URI	http://schemas.dmtf.org/cimi/2/NetworkServiceTemplate	
Attribute	**Type**	**Description**
policies	*map*	*** TBD *** Format & requirements yet to be determined form NSMWG work
meterTemplates	*meterTemplates []*	A list of references to `MeterTemplates` that shall be used to create and connect a set of new `Meters` to the new `NetworkService`. Note that the attributes of the `MeterTemplate` may be specified rather than a reference to an existing `MeterTemplate` Resource.
eventLogTemplate	*ref*	A reference to an `EventLogTemplate` that shall be used to create and connect a new `EventLog` to the new `NetworkService`. Note that the attributes of the `EventLogTemplate` may be specified rather than a reference to an existing `EventLogTemplate` Resource.

When implementing or using `NetworkServiceTemplate` Resources, Providers and Consumers shall adhere to the syntax and semantics of its attributes as described in Table 45 as well as in the tables describing embedded Resources or related Collections.

5.16.19.1 Collections

The `NetworkServiceTemplate`.Resource has no attributes of type Collection.

5.16.19.2 Operations

The `NetworkServiceTemplate` Resource supports the Read, Update, and Delete operations. Create is supported through the `NetworkServiceTemplateCollection` Resource.

5.16.20 NetworkServiceTemplateCollection Resource

A `NetworkServiceTemplateCollection` Resource represents the Collection of `NetworkServiceTemplates` within a Provider and follows the Collection pattern defined in clause 5.5.12. Operations

The `NetworkServiceTemplateCollection` Resource supports the Read and Update operations. Creation of new `NetworkServiceTemplate` Resources is supported by the way of a POST to the "add" operation's URI as described in clause 4.2.1.1.

5.16.21 Policies

****** TBD ******

Format & requirements yet to be determined form NSMWG work*Error! Reference source not found.***.**

5.17 Monitoring Resources and relationships

5.17.1 Job Resource

This Resource represents a process (i.e., a sequence of one or more operations directed to accomplish a specific goal) that is performed by the Provider.

If a Provider supports exposing `Job` Resources to Consumers, each request from a Consumer that the Provider responds to with a 202 status code, shall result in a `Job` Resource being created and an absolute URI reference to that `Job` Resource shall be made available to the requesting Consumer.

Providers may create additional Job Resources for Provider-initiated operations if the Provider chooses to expose these Jobs to Consumers.

If a Job is not completed successfully (e.g., it is in the FAILED or STOPPED state), this specification does not place any requirements on the Provider to ensure that the affected Resources are left in certain states. Based on the environmental conditions at that time, the Provider might choose to "undo" any impact of the operation; simply halt processing; attempt some kind of "cleanup" action; or choose to do something else. However, Providers shall list all Resources impacted by the Job in the "affectedResources" attribute, thus allowing Consumers an opportunity to examine the state of each Resource themselves. In cases where a Resource has been deleted, references to that Resource shall not appear in the "affectedResources" attribute.

The Job Resource allows for nesting of Jobs. The determination of when a single operation is converted into multiple nested Jobs is out of scope of this specification. However, if there are nested Jobs, the top-most Job Resource shall report the overall status of all Jobs and shall only be in a "SUCCESS" state if all nested Jobs are also in "SUCCESS" state. If nested Jobs are created, there is no requirement for the top-most Job Resource to reference all affected Resources in its "affectedResources" attribute. The Consumer needs to traverse the entire set of nested Jobs to determine the complete list of Resources impacted by the Jobs.

Table 46 describes the Job attributes.

Table 46 – Job attributes

Name	Job	
Type URI	http://schemas.dmtf.org/cimi/2/Job	
Attribute	**Type**	**Description**
state	*string*	The state of the process associated with this operation. Allowed values are: **QUEUED**: Indicates that the operation has not yet begun processing. **RUNNING**: Indicates that the operation is still being executed. **FAILED**: Indicates that the operation failed to be completed successfully. **SUCCESS**: Indicates that the operation was successfully completed. **STOPPING**: Indicates that the operation is in the process of being stopped. **STOPPED**: Indicates that the operation was stopped before completion. The operations that result in transitions to the above defined states are defined in clause 5.17.1.1
targetResource	*ref*	A reference to the top-level Resource upon which the operation is being performed. Typically, this Resource would be the Resource on which the operation was invoked. Note that if an "add" Job is executed against a "Collection" Resource (e.g., MachineCollection), the targetResource attribute shall reference the Collection Resource as that is the Resource on which the operation was performed. Additionally, the newly created Resource shall appear in the "affectedResources" attribute.
affectedResources	*ref[]*	A list of references to Resources that have been impacted by this Job. Note that this list shall always contain the "targetResource" reference. Array item name: affectedResource
action	*URI*	A URI that indicates the type of action being performed.
returnCode	*integer*	The operation return code. The specific value is specific to the implementation. Values in the range of 0 to 9999 are reserved for use by this specification.
progress	*integer*	An integer value in the range 0 … 100 that indicates the progress of this Job. This value shall be 100 if the Job is no longer executing, regardless of the outcome.

Name	Job	
Type URI	http://schemas.dmtf.org/cimi/2/Job	
Attribute	**Type**	**Description**
statusMessage	*string*	A human-readable string that provides information about the operation. It is used to further qualify or provide additional information about the current status of the operation. For example, this attribute may indicate the reason why the operation failed, or whether the operation was cancelled by the Consumer or the Provider.
timeOfStatusChange	*dateTime*	A timestamp indicating the last time that the status of the operation changed.
parentJob	*ref*	A reference to the `Job` of which this Resource is a subordinate i.e. a nested job
nestedJobs	*ref[]*	An array of references to a set of subordinate `Job` Resources. Array item name: nestedJob

When implementing or using `Job`, Providers and Consumers shall adhere to the syntax and semantics of its attributes as described in Table 46 as well as in the tables describing referred Resources or related Collections.

5.17.1.1 Operations Resource

This Resource supports the Read, Update, and Delete operations. Deleting a `Job` that is in the "RUNNING" state shall be the equivalent of first stopping the `Job` and then deleting it. A request to delete a running `Job` that does not support the "stop" action shall fail.

The following custom operations are also defined:

stop

/link@rel: `http://schemas.dmtf.org/cimi/2/action/stop`

This operation shall stop a `Job`.

Input parameters: None.

Output parameters: None.

During the processing of this operation, the `Job` shall be in the "STOPPING" state.

Upon successful completion of this operation, the `Job` shall be in the "STOPPED" state.

HTTP protocol

To stop a `Job`, a POST is sent to the "http://schemas.dmtf.org/cimi/2/action/stop" URI of the `Job` where the HTTP request body shall be as described below.

JSON media type: application/json

JSON serialization:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/Action",
  "action": "http://schemas.dmtf.org/cimi/2/action/stop",
  "properties": { string: string, + } ?
  ...
}
```

XML media type: application/xml

XML serialization

```
<Action xmlns="http://schemas.dmtf.org/cimi/2">
  <action> http://schemas.dmtf.org/cimi/2/action/stop </action>
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties> ?
  <xs:any>*
</Action>
```

Upon successful processing of the request, the HTTP response body may be empty.

5.17.2 JobCollection Resource

A `JobCollection` Resource represents the Collection of `Jobs` within a Provider and follows the Collection pattern defined in clause 5.5.12.

5.17.3 Meter Resource

This Resource represents an available `Meter` of some property associated to a given Resource.

If a `Meter's` "targetResource" is deleted all `Meters` associated with that Resource shall also be deleted. In other words, deleting a Resource-specific `MetersCollection` (e.g., a `Machine's MetersCollection`) shall also result in the deletion of the `Meters` referenced from that Collection.

Table 47 describes the `Meter` attributes.

Table 47 – Meter attributes

Name	Meter	
Type URI	http://schemas.dmtf.org/cimi/2/Meter	
Attribute	**Type**	**Description**
targetResource	*ref*	A reference to the Resource to which the `Meter` is related.
aspect	*URI*	A unique identifier representing the aspect of the Resource being metered.
units	*string*	The name of the used units, e.g., kilobits per second, CPU usage percentage, etc.
sampleInterval	*integer*	The time between consecutive samples in seconds.
timeScope	*string*	The time scope to which this meter's value applies. Two possible values: "Point" indicates that the Meter applies to a point in time. "Interval" indicates that the `Meter` applies to a time interval. For instance, it would be possible to define a `Meter` whose purpose is to provide the daily average CPU usage.
intervalDuration	*duration*	The interval duration when the timeScope is set to "Interval". Possible values: hourly, daily, weekly, monthly, or yearly.
isContinuous	*boolean*	This value indicates whether the `Meter` value is continuous or scalar. Performance Meters are an example of a linear metric.
samples	*collection [Sample]*	A reference to the list of taken samples

Name	Meter	
Type URI	http://schemas.dmtf.org/cimi/2/Meter	
Attribute	**Type**	**Description**
minValue	*string*	The expected minimal measure value.
maxValue	*string*	The expected maximum measure value.
stopTime	*dateTime*	The time from which the meter stops tracking samples.
expiresTime	*dateTime*	The time from which the `Meter` is not monitored anymore. It implies the deletion of the Meter after this time. Note that a `Meter` might be deleted before this time if the Resource being metered is deleted.

5.17.3.1 Collections

When implementing or using `Meter`, Providers and Consumers shall adhere to the syntax and semantics of its attributes as described in Table 47 as well as in the tables describing related Collections.

The following clauses describe the Collection resources that are components of `Meters`.

5.17.3.1.1 SampleCollection Resource

The Resource type for each item of this Collection is “`Sample`”, defined in Table 48:

Table 48 – Sample attributes

Name	Sample	
Type URI	http://schemas.dmtf.org/cimi/2/Sample	
Attribute	**Type**	**Description**
timestamp	*dateTime*	Indicates when the measure was taken (timeScope="Point"). If the timeScope is "Interval", it indicates the end of the time interval.
value	*string*	Indicates the sampled value of the measure.

5.17.3.2 Operations

When implementing or using `Sample`, Providers and Consumers shall adhere to the syntax and semantics of its attributes as described in Table 48 as well as in the tables describing related Collections.

This Resource supports the Read, Update, and Delete operations. Create is supported via the `MeterCollection` Resource. The deletion of a Meter shall remove the `Meter` from the targetResource's "meter" attribute.

The following custom operations are also defined:

start

/link@rel: `http://schemas.dmtf.org/cimi/2/action/start`

This operation shall start a `Meter`.

Input parameters: None.

Output parameters: None.

Upon successful completion of this operation, the `Meter` shall start recording samples related to its associated Resource.

HTTP protocol

To start a `Meter`, a POST is sent to the "http://schemas.dmtf.org/cimi/2/action/start" URI of the `Meter` where the HTTP request body shall be as described below.

JSON media type: application/json

JSON serialization:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/Action",
  "action": "http://schemas.dmtf.org/cimi/2/action/start",
  "properties": { string: string, + } ?
  ...
}
```

XML media type: application/xml

XML serialization

```
<Action xmlns="http://schemas.dmtf.org/cimi/2">
  <action> http://schemas.dmtf.org/cimi/2/action/start </action>
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties> ?
  <xs:any>*
</Action>
```

Upon successful processing of the request, the HTTP response body may be empty.

stop

/link@rel: `http://schemas.dmtf.org/cimi/2/action/stop`

This operation shall stop a `Meter`.

Input parameters: None.

Output parameters: None.

Upon successful completion of this operation, the `Meter` shall no longer be recording samples related to its associated Resource.

HTTP protocol

To stop a `Meter`, a POST is sent to the "http://schemas.dmtf.org/cimi/2/action/stop" URI of the `Meter` where the HTTP request body shall be as described below.

JSON media type: application/json

JSON serialization:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/2/Action",
  "action": "http://schemas.dmtf.org/cimi/2/action/stop",
  "properties": { string: string, + } ?
  ...
}
```

XML media type: application/xml

XML serialization

```
<Action xmlns="http://schemas.dmtf.org/cimi/2">
  <action> http://schemas.dmtf.org/cimi/2/action/stop </action>
  <properties>
    <property key="xs:string"> xs:string </property> *
  </properties> ?
  <xs:any>*
</Action>
```

Upon successful processing of the request, the HTTP response body may be empty.

5.17.4 MeterCollection Resource

A MeterCollection Resource represents the Collection of Meters within a Provider and follows the Collection pattern defined in clause 5.5.12.

5.17.4.1 Operations

NOTE The "add" operation requires that a MeterTemplate be used (see 4.2.1.1).

If Meters are created through the global (Cloud Entry Point) MeterCollection's "add" operation, they shall be added automatically to the corresponding targetResource's "Meters" Collection Resource as well.

5.17.5 MeterTemplate Resource

A MeterTemplate represents the information needed to create a new Meter. Table 49 describes the MeterTemplate attributes.

Table 49 – MeterTemplate attributes

Name	MeterTemplate	
Type URI	http://schemas.dmtf.org/cimi/2/MeterTemplate	
Attribute	**Type**	**Description**
targetResource	*ref*	A reference to the Resource that is metered. The type of the Resource shall be one of the "associatedTo" types listed in the MeterConfiguration referenced. If this Template is used to create a new Meter through the global (Cloud Entry Point) MetersCollection, this attribute shall be present. If this Template is used to create a new Meter through a targetResource's MetersCollection, this attribute shall either be absent or have the same value as the "id" of the targetResource to which this Meter is being added.
meterConfig	*ref*	A reference to the MeterConfiguration that is used to create a Meter from this MeterTemplate. Note that the attributes of the MeterConfiguration may be specified rather than a reference to an existing MeterConfiguration Resource.

When implementing or using `MeterTemplate`, Providers and Consumers shall adhere to the syntax and semantics of its attributes as described in **Table 49** as well as in the tables describing referred Resources or related Collections.

5.17.6 MeterTemplateCollection Resource

A `MeterTemplateCollection` Resource represents the Collection of `MeterTemplate` Resources within a Provider and follows the Collection pattern defined in clause 5.5.12.

5.17.6.1 Operations

This Resource supports the Read and Update operations. Creation of new `MeterTemplate` Resources is supported by the way of a POST to the "add" operation's URI as described in clause 4.2.1.1.

5.17.7 MeterConfiguration Resource

A `MeterConfiguration` represents the definition of a `Meter`. Table 50 describes the `MeterConfiguration` attributes.

Table 50 – MeterConfiguration attributes

Name	MeterConfiguration	
Type URI	http://schemas.dmtf.org/cimi/2/MeterConfiguration	
Attribute	**Type**	**Description**
associatedResources	*URI[]*	An array of URIs that indicate the types of Resources to which a `Meter` created from this configuration can be applied. The value space of these URIs is identical to that of `ResourceMetadata`.typeURI, which is a URI that uniquely identifies a Resource type.
aspect	*URI*	A unique identifier representing the aspect of the Resource being metered. See Table 51 below for the set of CIMI-defined URIs.
units	*string*	The human-readable name of the used units, e.g., kilobits per second, CPU usage percentage, etc.
sampleInterval	*integer*	The time between consecutive samples in seconds.
timeScope	*string*	The time scope to which the `Meter` value applies. Two possible values: "Point" indicates that the `Meter` applies to a point in time. "Interval" indicates that the `Meter` applies to a time interval. For instance, it would be possible to define a `MeterConfiguration` whose purpose is to provide the daily average CPU usage.
intervalDuration	*duration*	The interval duration when the timeScope is set to "Interval." Possible values: hourly, daily, weekly, monthly, or yearly.
isContinuous	*boolean*	This value indicates whether the `Meter` value is continuous or scalar. Performance `Meters` are an example of a linear metric.

Table 51 describes the "aspect" URIs defined by this specification. Providers may define new aspect URIs and it is recommended that these URIs be dereferencable such that Consumers can discover the details of the new aspect. For brevity the "URI" column in the table only shows the last part of the URI. It should be appended to: "http://schemas.dmtf.org/cimi/2/aspect/".

Table 51 – aspect URIs

Aspect	Description
cpu	The percentage CPU usage of the Resource. Typically associated with `CloudEntryPoint`, `System`, and `Machine` Resources. For Resources that group other Resources (e.g., `CloudEntryPoint` or System `Resources`), this aspect provides the aggregated percentage usage of the CPU.
memory	The amount of memory being used by the Resource. Typically associated with `CloudEntryPoint`, `System`, and `Machine` Resources. For Resources that group other Resources (e.g., `CloudEntryPoint` or `System` Resources), this aspect provides the aggregated usage of the memory.
disk	The amount of disk being used by the Resource. Typically associated with `CloudEntryPoint`, `System`, `Machine`, and `Volume` Resources. For Resources that group other Resources (e.g., `CloudEntryPoint` or `System` Resources), this aspect provides the aggregated disk usage.
bandwidth	The amount of network traffic. Typically associated with `CloudEntryPoint`, `System`, and `Network` Resources. For `CloudEntryPoint` and `System` Resources, this aspect provides the aggregated bandwidth of all the networks under them.
inputBandwidth	The amount of input bandwidth used by the Resource. Typically associated with `Machine`, `NetworkPort`, and `Volume` Resources. For `Machine` Resources, this aspect provides the aggregated input bandwidth usage of all its network interfaces .
outputBandwidth	The amount of output bandwidth used by the Resource. Typically associated with `Machine`, `NetworkPort`, and `Volume` Resources. For `Machine` Resources, this aspect provides the aggregated output bandwidth usage of all its network interfaces.

5.17.7.1 Operations

This Resource supports the Read, Update, and Delete operations. Create is supported through the `MeterConfigurationCollection` Resource.

5.17.8 EventLog Resource

A Resource that represents a registry of Events.

If an `EventLog's` "targetResource" is deleted the `EventLog` associated with that Resource may also be deleted. In other words, deleting a Resource (e.g., a `Machine`) may also result in the deletion of the `EventLog` referenced from that Resource. This behavior is denoted by the `EventLog` ”Linked” capability.

If an EventLog is deleted, all of its Events shall also be deleted.

5.17.9 MeterConfigurationCollection Resource

A MeterConfigurationCollection Resource represents the Collection of MeterConfigurations within a Provider and follows the Collection pattern defined in clause 5.5.12.

5.17.9.1 Operations

This Resource supports the Read and Update operations. Creation of new MeterConfiguration Resources is supported by the way of a POST to the "add" operation's URI as described in clause 4.2.1.1.

Table 52 describes the `EventLog` attributes.

5.17.10 MeterConfigurationCollection Resource

A `MeterConfigurationCollection` Resource represents the Collection of `MeterConfigurations` within a Provider and follows the Collection pattern defined in clause 5.5.12.

5.17.10.1 Operations

This Resource supports the Read and Update operations. Creation of new `MeterConfiguration` Resources is supported by the way of a POST to the "add" operation's URI as described in clause 4.2.1.1.

Table 52 – EventLog attributes

Name	EventLog	
Type URI	http://schemas.dmtf.org/cimi/2/EventLog	
Attribute	**Type**	**Description**
targetResource	*ref*	A reference to the Resource to which the `Events` are related.
Events	*collection [Event]*	A reference to the list of occurred `Events`.
Persistence	*string*	A value that indicates the persistence of the `Events` within the `EventLog`. For instance, daily, weekly, monthly, or yearly. Events that exceed the persistence duration may be deleted.
Summary	*<unnamed structure>*	A summary of all the events present in the `EventLog` when the read operation is performed, grouped by severity. Each summary attribute is an (unnamed) structure that has the following sub-attributes:

Attribute	**Type**	**Description**
low	*integer*	Number of occurred `Events` with a low severity.
medium	*integer*	Number of occurred `Events` with a medium severity.
high	*integer*	Number of occurred `Events` with a high severity.
critical	*integer*	Number of occurred `Events` with a critical severity.

When implementing or using `EventLog`, Providers and Consumers shall adhere to the syntax and semantics of its attributes as described in `MeterConfigurationCollection` Resource.

A `MeterConfigurationCollection` Resource represents the Collection of `MeterConfigurations` within a Provider and follows the Collection pattern defined in clause 5.5.12.

5.17.10.2 Operations

This Resource supports the Read and Update operations. Creation of new MeterConfiguration Resources is supported by the way of a POST to the "add" operation's URI as described in clause 4.2.1.1.
MeterConfigurationCollection Resource

A `MeterConfigurationCollection` Resource represents the Collection of `MeterConfigurations` within a Provider and follows the Collection pattern defined in clause 5.5.12.

5.17.10.3 Operations

This Resource supports the Read and Update operations. Creation of new `MeterConfiguration` Resources is supported by the way of a POST to the "add" operation's URI as described in clause 4.2.1.1.

Table 52 Table 52 as well as in the tables describing embedded Resources or related Collections.

5.17.10.4 Collections

The following clauses describe the Collection Resources EventLogs.

5.17.10.4.1events Collection

The Resource type for each item of this Collection is "`Event`" as defined in clause 5.17.14.

5.17.10.5 Operations

This Resource supports the Read, Update, and Delete operations.

5.17.11 EventLogCollection Resource

An `EventLogCollection` Resource represents the Collection of `EventLogs` within a Provider and follows the Collection pattern defined in clause 5.5.12.

5.17.12 EventLogTemplate Resource

An `EventLogTemplate` represents the information needed to create a new `EventLog`. Table 53 describes the `EventLogTemplate` attributes.

Table 53 – EventLogTemplate attributes

Name	EventLogTemplate	
Type URI	http://schemas.dmtf.org/cimi/2/EventLogTemplate	
Attribute	**Type**	**Description**
targetResource	*ref*	A reference to the Resource to which the EventLog shall be connected.
persistence	*string*	A value that indicates the persistence of the Events in the new `EventLog`. For instance, daily, weekly, monthly, or yearly. `Events` that exceed the persistence duration may be deleted.

When implementing or using `EventLogTemplate`, Providers and Consumers shall adhere to the syntax and semantics of its attributes as described in **Table 53** as well as in the tables describing referred Resources or related Collections.

5.17.13 EventLogTemplateCollection Resource

An `EventLogTemplateCollection` Resource represents the Collection of `EventLogTemplate` Resources within a Provider and follows the Collection pattern defined in clause 5.5.12.

5.17.13.1 Operations

This Resource supports the Read and Update operations. Creation of new `EventLogTemplate` Resources is supported by the way of a POST to the "add" operation's URI as described in clause 4.2.1.1.

5.17.14 Event Resource

A Resource that represents the occurrence of an event within the managed infrastructure. Some examples of `Event` are:

- Machine X has been rebooted by guest OS.

- Machine X is not responding to platform services.
- A new vCPU has been added to machine X following defined elasticity rules.

The scope of the Event concept is any information that the Provider is able to track within its infrastructure and that can constitute useful information for the Consumer. Possible examples include, but are not limited to, errors and inconveniences that occur in the (virtual) resources assigned to Consumers; Provider-initiated actions, such as maintenance tasks; etc.

Table 54 describes the Event attributes.

Table 54 – Event attributes

Name	Event	
Type URI	http://schemas.dmtf.org/cimi/2/Event	
Attribute	**Type**	**Description**
timestamp	*dateTime*	The time of occurrence of the actual Event. NOTE: This attribute should not be confused with the time of creation of the Event Resource instance, which is captured in the common "created" attribute.
type	*URI*	A URI that uniquely identifies the type of the Event. If the "content" attribute is present, this URI determines the actual data structure used for this content, e.g., to which schema it is associated.
content	*any*	A polymorphic attribute that represents detailed event data, the type of which varies with the Event "type." Typically, a data structure; for example: In the case of a monitoring event, the content shall hold the target Resource ID and type, measured attribute(s), and status value(s). In the case of an audit event conforming to the CADF model, the content shall hold the detailed event structure that complies with CADF event schema. In the case of a CIM Indication, the content shall hold the structure and attributes defined for such events.
outcome	*string*	A string value that characterizes the general significance of the Event. A core set is defined that may be used regardless of the Event type. For each Event type, the definition of a core outcome value maybe refined in the context of this type, provided it does not conflict with the general meaning of the outcome given below. Core outcomes are: **Pending**: The Event is about an action or process that is still ongoing. **Unknown**: The Event is about a request or action that is not known by the Provider. **Status**: The Event reports on the state or status of a Resource. **Success**: The Event reports on a successful outcome of some action or process. **Warning**: The Event reports on a situation that requires attention or remedial action. **Failure**: The Event reports on a failed outcome of some action or process. This set of core outcome values may be extended to accommodate possible outcomes of a specific Event type. In this case, the extended set of values shall apply to all Events of this type.
severity	*string*	A value indicating the Event severity. Possible values are: **critical** **high** **medium** **low** The meaning of the severity level may vary depending on the Event "type." If such an attribute is not relevant to a particular type of Event, it should be omitted.
contact	*string*	A reference to a contact point or processing point to handle the Event. The actual type of this content (e.g., email address, phone number of helpdesk or staff, message queue, URL…) is dependent on, and determined by the Event "type." This attribute is mutable as it

Name	Event	
Type URI	http://schemas.dmtf.org/cimi/2/Event	
Attribute	**Type**	**Description**
		may be determined after `Event` creation by the Provider.

NOTE There exists a legacy of several `Event` models that have been standardized or designed for various domains relevant to IT. The objective in CIMI is not to elect one particular `Event` model, but to select as top-level `Event` attributes the most immediately relevant data useful for `Event` processing in a Cloud environment. Additional `Event` data may still be represented in the variable content attribute that allows for mapping other `Event` models into a CIMI `Event`.

When implementing or using `Event`, Providers and Consumers shall adhere to the syntax and semantics of its attributes as described in Table 54.

Table 55 describes the "type" URIs that are defined or acknowledged by this specification. Additional types may be added by a Provider, for example to characterize external events mapped into CIMI `Events`. It is recommended that these URIs be dereferencable such that Consumers can discover a more detailed description of the type. `Event` types defined by this specification share the same base URI: http://schemas.dmtf.org/cimi/2/event/. For brevity, if the "Event Type" column in the table only shows a relative URI (e.g., state) it shall be appended to the end of this base URI.

Table 55 – type URIs

Event Type	Description
state	Events of this type report state information about CIMI run-time resources such as instances of Machines, Systems, Networks, and Volumes. This information includes reports on any change in the "state" of these Resources. The **content** element associated with this `Event` type has the following structure:

Data	Type	Description
resName	*string*	The name of the Resource about the state of which is reported.
resource	*ref*	The reference to the Resource about the state of which is reported. (Note: This reference may become invalid because the event might outlive the Resource.)
resType	*URI*	URI denoting this Resource type (same as the type URI associated with the Resource type for this Resource).
state	*string*	The state reported for the Resource. Shall be the same as the "state" attribute value (if any) of the run-time Resource at the time the event is generated.
previous	*string*	The previous state value, if the event reports a state change.

Event Type	Description
alarm	`Events` of this type report errors or alarms occurring during management operations of Cloud resources. This information includes failures to provision resources, failures to fulfill requests to the CIMI interface, and any critical situation that needs be addressed in a timely manner. The **content** element associated with this event type has the following structure:

Data	Type	Description
resName	*string*	The name of the Resource associated with this alarm, if applicable.
resource	*ref*	The reference to the Resource associated with this alarm, if applicable. (Note: This reference may become invalid because the event might outlive the Resource.)
restype	*URI*	URI denoting this Resource type associated with this alarm, if applicable (same as the type URI associated with the Resource type for this Resource).
code	*string*	An alarm code.
detail	*string*	The detailed information associated with the alarm.

Event Type	Description
model	`Events` of this type report changes in the CIMI resource model, which includes creation, modification, and destruction of Resource instances; and updates to metadata (Resource extensions, capabilities and constraints, etc.). The **content** element associated with this event type has the following structure:

Data	Type	Description
resName	*string*	The name of the main model Resource affected by the modification.
resource	*ref*	The reference to the main model Resource affected by the modification. (Note: This reference may become invalid because the event might outlive the Resource.)
resType	*URI*	URI denoting this Resource type (same as the type URI associated with the Resource type for this Resource).
change	*string*	The kind of modification reported (create/update/delete).
detail	*string*	The detailed information associated with the change, typically the data for an update or creation, as used in a request.

Event Type	Description
access	`Events` of this type keep track of all requests to access some Resource of a CIMI provider. The **content** element associated with this event type has the following structure:

Data	Type	Description
operation	*string*	The method or name of the operation intended for this access (for the HTTP protocol, the HTTP method for the request).
resource	*ref*	The reference of the Resource supporting the operation (for the HTTP protocol, the Resource URI or the URI associated with the operation). (Note: This reference may become invalid because the event might outlive the Resource.)
detail	*string*	The detailed information associated with the change, typically the data for an update or creation, as used in a request
initiator	*string*	The details identifying the request initiator, in case that information can be associated with the request.

Event Type	Description
http://schemas.dmtf.org/cloud/audit/1.0/	Events of this type represent events that have audit significance, as defined by CADF (…). This type can be subdivided further by extending the URI path (e.g., http://schemas.dmtf.org/cloud/audit/1.0/event/security, for security audit events). The **content** element associated with this event type has the same structure as the event serialization defined in CADF (DSP0262)

5.17.14.1 Operations

This resource supports the Read, Update, and Delete operations.

6 Security considerations

There are many security mechanisms that can be used in conjunction with this specification. This specification does not mandate any particular mechanism. Providers shall provide enough information about their security mechanisms so that the Consumer can implement the necessary algorithms to successfully communicate with the Provider.

An implementation may set limits on the length of attribute values it accepts. An implementation may set limits on the size of arrays it accepts. An implementation may set limits on the size of the request body or the length of request URIs it accepts. These limits may not all be advertised in the ResourceMetadata, although this specification recommends Providers to do so. A Provider that receives a request that exceeds any of these limits, shall return a response with an appropriate standard HTTP status code.

7 Conformance

This section decribes a mimimal set of features that a Cloud Provider must implement to be in conformance with the specification.

This does not preclude a implementing additional features and is not exclusive of other levels of conformance that may be defined outside of this document.

The goal is to specify a basic set of features upon which implementions may rely that provides useful functionality and aids interoperability without making onerous demands on Cloud Provider implementations.

7.1 Minimal Conformance Clause

A Cloud Provider implementation is in minimal conformance with the specification if it satisfies all of the following requirements:

- It implements the **Machine** Resource specified in section 5.14 “Machine Resources and Relationships”, along with its mandatory (providerMandatory=true) common attributes, and at least the following attributes: `cpu, memory, disks, cpuArch, cpuSpeed`,
- It implements the **MachineImage** Resource specified in section 5.14.7 “MachineImage Resource”, along with its mandatory (providerMandatory=true) common attributes, and at least the following attributes: `imageLocation`.,
- It implements the **MachineConfiguration** Resource specified in section 5.14.5 “MachineConfiguration Resource” along with its mandatory (providerMandatory=true) common attributes, and at least the following attributes: `cpu, memory, disks, cpuArch, cpuSpeed`, in addition to mandatory common attributes,
- It implements **ResourceMetadata** ResourceMetadata specified in section 5.11 “Resource Metadata”, with at least the attributes: `typeURI, name, attributes`, and all the fields in the *attribute* data type except for `consumerMandatory`*.* The minimal support required for ResourceMetadata is only for discovery via the CEP. No access is required from any other Resource i.e. no ResourceMetadata reference is required in any other Resource.
- It supports the creation of Machine Resources with template data passed by value, as specified in section 4.2.1.1 “Creating a new Resource”, i.e. is able to process a Machine creation request where the Machine template is passed by value. No support for the MachineTemplate Resource is required.
- It implements the **Collection** Resource as specified in section 5.5.12 “Collection” for the following Resources: ResourceMetadata, Machine, MachineImage, MachineConfiguration, as specified in section 5.14.2 “MachineCollection Resource”, section 5.14.6 “MachineConfigurationCollection Resource”, section 5.14.8 “MachineImageCollection Resource” and section 5.11.2 “Resource MetadataCollection Resources”,
- It implements the **CEP** Resource as specified in section 5.12 “Cloud Entry Point” , with the following collection attributes: resourceMetadata, machines, machineImages, machineConfigs,
- For all the above Resources, it provides at least read-only access to their attributes, and at least the create and delete operations.
- It handles requests and generates responses according to protocol requirements as specified in section 4.2 “Protocol operations”.
- It handles content serialization in requests and serializes content in generated messages as specified in section 5.5 “Data types and their serialization”.

ANNEX A OVF support in CIMI

This annex defines how elements of an OVF descriptor are mapped to CIMI resources and their attributes. This definition allows the import of an OVF package to create multiple CIMI resources. This is done by specifying a reference to an OVF package in the import operation of a `SystemCollection` or `SystemTemplateCollection` (the Media Type at that URI shall be "application/ovf"). Refer to DSP0243 for more information about OVF.

Support for OVF import and export is optional for a Provider and it is an implementation choice as to how many of the attributes in the OVF package are exposed through CIMI resources. A Provider may support the import of OVF package for only `Systems`, only `SystemTemplates` or both. Support for the actual import and export of an OVF package is handled by a hypervisor under the management of the CIMI implementation, and thus the CIMI resources that are created reflect what the hypervisor did upon import and form a "View" into the results.

The import of an OVF package can be reflected in the creation of Templates that can be later used to create `Systems`, `Machines` and other component Resources. The import of an OVF package can also be used to directly create `Systems`, `Machines`, and other component Resources, bypassing the step of creating Templates.

Clause 5.13.5 details how to import an OVF file to create a `SystemTemplate` (and component Resources). The `SystemTemplate` thus created contains a reference to a `MachineTemplate` for every `VirtualSystem` that is defined in the OVF descriptor `VirtualSystemCollection`. Note that CIMI currently allows `Systems` of `Systems`, so for each `VirtualSystemCollection` encountered in a nested set of collections, a separate `SystemTemplate` is created within the parent `SystemTemplate` with `MachineTemplates` for each of the contained `VirtualSystems` in that `VirtualSystemCollection`.

The values of the attributes for the `MachineTemplate` are taken from the `VirtualHardwareSection` of the `VirtualSystem` description (required in OVF). If more than one `VirtualHardwareSection` is used for a given `VirtualSystem` (allowed in OVF), the result is implementation dependent, but the implementation might choose a `MachineTemplate` from an existing (perhaps static) set that best matches a `VirtualHardwareSection`. Items in the `VirtualHardwareSection` are mapped to CIMI `MachineConfiguration` properties and the corresponding `MachineConfiguration` Resource is created and linked to from the created `MachineTemplate` for that `VirtualSystem`.

The CIMI `VolumeTemplates` are created according to the `DiskSection` of an OVF descriptor and can be shared among more than one `VirtualSystem` (CIMI `MachineTemplates`) defined in an OVF package. In addition, a new CIMI `MachineImage` Resource may be created from the `DiskSection` if an `ovf:fileRef` for the virtual disk content is specified.

The CIMI `NetworkTemplates` are created according to the `NetworkSection` of an OVF descriptor along with the `Connection` elements in the `VirtualHardwareSection` elements that refer to these named networks.

Clause 0 details how to import an OVF file to create a `System` (and component Resources). The `System` thus created contains a reference to a `Machine` for every `VirtualSystem` that is defined in an OVF descriptor `VirtualSystemCollection`. Note that CIMI currently allows `Systems` of `Systems`, so for each `VirtualSystemCollection` encountered in a nested set of collections, a separate System is created within the parent `System` with `Machines` for each of the contained `VirtualSystems` in that `VirtualSystemCollection`.

The values of the attributes for the `Machine` are taken from the `VirtualHardwareSection` of the `VirtualSystem` description (required in OVF). If more than one `VirtualHardwareSection` is used for a given `VirtualSystem` (allowed in OVF), the result is implementation dependent. Items in the `VirtualHardwareSection` are mapped to CIMI `MachineConfiguration` properties and the corresponding `MachineConfiguration` Resource is created and linked to from the created `Machine` for that VirtualSystem.

The CIMI `Volumes` are created according to the `DiskSection` of an OVF descriptor and can be shared among more than one `VirtualSystem` (CIMI `Machines`) defined in an OVF package. In addition, a new CIMI `MachineImage` Resource may be created from the `DiskSection` if an `ovf:fileRef` attribute for the virtual disk content is specified.

The CIMI `Networks` are created according to the `NetworkSection` of an OVF descriptor along with the `Connection` elements in the `VirtualHardwareSection` that refer to these named networks.

ANNEX B XML Schema

The XML Schema for the XML serialization of the CIMI model can be found at:

http://schemas.dmtf.org/cimi/2/dsp8009_1.0.xsd

The schema provided does not intend to reflect every single modeling constraint and requirement specified in the model. This schema is designed to apply more broadly to any model-related serialized material found in Consumer requests as well as in Provider responses, and is intended to provide a preliminary, non-exhaustive syntactic check on these. In particular, future updates of this specification may intermix new XML elements into the Resources using the current CIMI namespace to Resources. The schema that is provided is just a starting point for those who would find it useful and it might need to be modified based on specific application's needs.

ANNEX C Change log

Version	Date	Who	Description
1.1.0a	08/13/2013	BrightLeaf	DMTF Draft Standard
0.0.126	10/22/2013	Jacques	Editorial changes to resolve mantis issue 2159
0.0.127	11/18/2013	Jacques	Editorial changes to resolve mantis issue 1455
0.0.128	12/17/2013	Jacques	Editorial changes to resolve mantis issue 2252
0.0.129	1/7/2014	Jacques	Part of mantis issue 1455 resolution (updates to section 5.11 – using ValueScope in ResourceMetadata) was not implemented by mistake in 0.128. Done in 0.129, see diffs in 5.11.
0.0.130	1/21/2014	Jacques	Resolving mantis issues 2300 (clarify ordering of entries in a collection) , 2301 (string sorting with $orderby), 2302 (clarify that a Collection's "count" holds the value of the collection after filtering but before subsetting) , 2265 Not enough information why an operation is unavailable. NOTE: still need the serialization extension to be reported on every concerned resource (in addition to fix in 4.2.
0.0.131	1/22/2014	Jacques	Improved rewording on issues 2300
0.0.132	1/22/2014	Jacques	Second Improved rewording on issues 2300
0.0.133	1/22/2014	Jacques	Resolution of issue 2295 (Initial value of attribute "updated" not defined)
0.0.134	1/22/2014	Jacques	Resolution of issue 2233 (Improve the usability of the network template). Revision of resolution for issues 2300
0.0.135	1/23/2014	Jacques	revised resolutions of issues 2311, 2265, and Resolutions or 2307
0.0.136	1/23/2014	Jacques	revised resolution of issue 2311,
0.0.137	2/3/2014	Jacques	Resolution of issues: 2303 (explain dollar in CIMI query parameters), 2310 (allow Providers to limit entries in a returned collection when number of entries would be excessive) and 2313 (processing order of sorting and collection subsetting)
0.0.138	3/4/2014	Jacques	Resolution of issues: 2278: missing states and operations in Machine and Volume for capture/snapshot and restore, 2314: Section 5.11.1 "Serialization of attribute value constraints" should be removed
0.0.139	5/12/2014	Jacques	Resolution of issues: 2095: “providers should be allowed to refuse unreasonable requests”, and 2275: “typo in 1.1: xml schema url, etc”.
0.0.140	7/8/2014	Jacques	Resolution of issues: 2447 “Redefinition of NetworkInterfaces”
0.0.141	8/4/2014	Jacques	Resolution of issues: 2374 “using Collections for 1-to-many connections between existing resources is too complex”, 2442 “Incorrect use of networkPort vs. networkNetworkPort in CIMI”, 2436 “NetworkPort creation from the Network”, 2240 “Requirement for Job support should be narrowed to asynchronous responses.”, 2447 “Redefinition of NetworkInterfaces”.
0.0.142	8/12/2014	Jacques	Resolution of issues: 2448 “ResourceMetadata's attribute's type attribute is optional in JSON, mandatory in XML.” 2423 “Different treatment of the images of different Resources” - Additional fixing of cross-section referencing at several places for Collection sections (editor’s discretion)
0.0.143	9/19/2014	Jacques	Some rewording for: 2374 “using Collections for 1-to-many connections between existing resources is too complex”,
0.0.144	11/4/2014	Jacques	Resolution of issues:

Version	Date	Who	Description
			2393: SystemTemplate's importImage should be of type expRef, not xs:anyURI 2460: MachineTemplate uses NetworkInterface, Machine uses MachineNetworkInterface
0.0.145	16/2/2015	Jacques	Resolution of issues: 2527: Type URI for CloudEntryPoint is wrong 2521: CEP collections do not scale, do not help management of owned resources. 2535: New Network Model 2506: CIMI enhancements to address Business Continuity
0.0.146	20/2/2015	Jacques	Editorial scrub on 0.145, with: - Removal of misplaced and redundant examples from 5.5.12 - Fixed examples serialization in 5.14.2 - “Old” Network UML diagram removed and temporarily replaced with the new Network outline diagram proposed by Eric. Otherwise equivalent to 0.145, with all diffs removed, but remaining comments are left in.
0.0.147	20/4/2015	Jacques	Resolution of issues: 2564: XML serialization of arrays and lists of elements is lacking container element. 2566: Templates need be able to refer to and integrate "run-time" Resources.
0.0.148	26/5/2015	Jacques	Resolution of issues: 2565: ResourceMetadata scope needs to apply per Resource as well. 2563: Need a simpler rule for what Resources show in CEP collections (latest proposal: reverse to the original single option of CEP=exhaustive catalog of resources.)
0.0.149	28/5/2015	Jacques	Completed resolution of issue: 2565: ResourceMetadata scope needs to apply per Resource as well. - Added definitions of the new “attribute constraints” nomenclature, in section 5.3 (replacing the old ones) - Converted the remaining cases of old “attribute constraints” into the new proposed constraints nomenclature.
0.0.150	22/6/2015	Jacques	Updated resolution of issue: 2565: ResourceMetadata scope needs to apply per Resource as well. - Reworded some of the constraints definitions (5.3) based on feedback in recent meeting 6/2. Editorial: Grouped serialization rules in contiguous sections (i.e. 5.4, 5.5). Formerly 5.1 moved to 5.4 and is retitled “Serialization of Resources”, it also introduces section 5.5. (this is in anticipation of #2570)
0.0.151	26/10/2015	Jacques	Resolution of issues: • #2570: Only JSON and XML serialization rules should remain in DSP0263,, the actual Resources serializations should be inPrimer • #2586: Need a more generic definition for SystemServices, and clarify relationship to NetworkService if any. • #2593: Operation to connect two resources may require more than just inserting a Resource reference into another

Version	Date	Who	Description
			Resource's collection
0.0.152	4/1/2016	Jacques	Resolution of issues: • #2592: Conformance clause missng for CIMI. • #2549: UML diagrams for groups of resources are not up-to-date
0.0.153	27/1/2016	Jacques	Resolution of issues: • #2617: "properties" common attribute appears to be redundant with resourceMetadata • #2618: "Job" Resource has redundant parentJob attribute • #2619: outdated reference: RFC2616 which should be updated to RFC7230 • #2592: Conformance clause missng for CIMI.(reworded)
0.0.154	29/2/2016	Jacques	Resolution of editorial review comments, for next WIP:
2.0.0d	2016-03-22		Work in Progress version

Bibliography

DMTF Standard: *Cloud Infrastructure Management Interface (CIMI) Model and RESTful HTTP-based Protocol* specification V1.0 (DSP0263)

http://dmtf.org/sites/default/files/standards/documents/DSP0263_1.0.0.pdf

DMTF Standard: *Cloud Infrastructure Management Interface (CIMI) Model and RESTful HTTP-based Protocol* specification V1.1 (DSP0263)

https://members.dmtf.org/apps/org/workgroup/cmwg/download.php/73648/DSP0263_1.1.0b_RC2.pdf