WHITE PAPER

OVF
Open Virtual Machine Format Specification

version 0.9

VMware, Inc. 3401 Hillview Ave., Palo Alto CA, 94304 USA Tel 877-486-9273 Fax 650-427-5001 www.vmware.com
XenSource, Inc. 2300 Geng Road, Suite 2500, Palo Alto, CA 94303 www.xensource.com

68

69 **Document number: DSP0xxx**

70 **Date: 2007-09-07**

71 **Version: 0.0.0**

72 Open Virtual Machine Format Specification
73 (OVF)

74 **Document type: Specification**

75 **Document status: Draft**

76 **Document language: E**

Contents

1 Foreword 5
2 Introduction 6
3 Scope 7
3.1 Requirements 7
3.2 Notations and Terminology 7
3.3 Notational Conventions 7
4 Conformance 7
4.1 XML Namespaces 8
5 Terms and Definitions 8
6 Package Contents 10
6.1 File Structure 10
6.2 Virtual Disk Formats 11
6.3 Distribution 11
7 OVF Descriptor 11
8 OVF Envelope 12
9 Virtual Hardware Description 14
10 Core Meta-Data Sections 17
10.1 DiskSection 18
10.2 NetworkSection 18
10.3 ResourceAllocationSection 18
10.4 AnnotationSection 19
10.5 ProductSection 19
10.6 PropertySection 19
10.7 EulaSection 20
10.8 StartupSection 21
10.9 CpuCompatibilitySection 21
10.10 OperatingSystemSection 21
10.11 InstallationSection 21
11 OVF Environment 22
11.1 Environment Document Protocol 22
11.2 Installation Feedback Protocol 23
11.3 Transport 23
ANNEX A: Change Log 24
ANNEX B: OVF XSD 25
Schema: ovf-envelope.xsd 25
Schema: ovf-section.xsd 27
Schema: ovf-core.xsd 28
Schema: ovf-virtualhardware.xsd 33
Schema: cim-rasd.xsd 34
Schema: cim-vssd.xsd 38
Schema: cim-common.xsd 39
Schema: ovf-environment.xsd 42
MOF: CIM_VirtualSystemSettingData 44
MOF: CIM_ResourceAllocationSettingData 45
ANNEX C: Authors and Acknowledgements 48
Bibliography 49

1 Foreword

This will be added by the DMTF working group.

2 Introduction

The Open Virtual Machine Format (OVF) describes an open, secure, portable, efficient and extensible format for the packaging and distribution of (collections of) virtual machines. The key properties of the format are:

- **Optimized for distribution**
  Supports content verification and integrity checking based on industry standard public key infrastructure, and provides a basic scheme for management of software licensing.

- **Optimized for a simple, automated user experience**
  Supports validation of the entire package and each virtual machine or meta-data component of the OVF during the installation phases of the VM lifecycle management process. It also packages with the appliance relevant user-readable descriptive information that can be use by a virtualization platform to streamline the installation experience.

- **Supports both single VM and multi-VM configurations**
  Supports both standard single VM packages, and packages containing complex, multi-tier services consisting of multiple interdependent VMs.

- **Portable VM packaging**
  OVF is virtualization platform neutral, while also enabling platform-specific enhancements to be captured. It supports the full range of virtual hard disk formats used for VMs today, and is extensible to deal with future formats that may arise. Virtual machine properties are captured concisely and accurately.

- **Vendor and platform independent**
  The OVF does not rely on the use of a specific host platform, virtualization platform, or guest operating system (within the appliance).

- **Extensible**
  OVF is immediately useful – and extensible. It is designed to be extended as the industry moves forward with the virtual appliance technology. It also supports and permits the encoding of vendor specific meta-data to support specific vertical markets.

- **Localizable**
  Supports user visible descriptions in multiple locales, and supports localization of the interactive processes during installation of an appliance. This allows a single packaged appliance to serve multiple market opportunities.

- **Open standard**
  The OVF has arisen from the collaboration of key vendors in the industry, and will be developed in an accepted industry forum as a future standard for portable virtual machines.

It is not an explicit goal for OVF to be an efficient execution format. A hypervisor is allowed but not required to run any packaged VM directly out of this format.

The following sections contain the specification of the Open Virtual Machine Format.

3 Scope

3.1 *Requirements*

This specification intends to meet the following requirements:

Facilitate the automated, secure management not only of virtual machines but the appliance as a functional unit.

3.2 *Notations and Terminology*

This section specifies the notations, namespaces, and terminology used in this specification.

3.3 *Notational Conventions*

In this document, the keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119 [RFC 2119].

This specification uses the following syntax to define normative outlines for messages.

The syntax appears as an XML instance, but values in italics indicate data types instead of values.

Characters are appended to elements and attributes to indicate cardinality:

"?" (0 or 1)

"*" (0 or more)

"+" (1 or more)

The character "|" indicates a choice between alternatives.

The characters "[" and "]" indicate that enclosed items are to be treated as a group with respect to cardinality or choice.

An ellipsis ("...") indicates a point of extensibility that allows other child or attribute content. Additional children and/or attributes MAY be added at the indicated extension points but MUST NOT contradict the semantics of the parent and/or owner, respectively. If a receiver does not recognize an extension, the receiver SHOULD NOT process the message and MAY fault.

XML namespace prefixes (see Table 1) indicate the namespace of the element being defined.

Throughout the document, whitespace within XML element values is used for readability. In practice, a service can accept and strip leading and trailing whitespace within element values as if whitespace had not been used. (See conformance rule R10.2-10.)

4 Conformance

An implementation is not conformant with this specification if it fails to satisfy one or more of the MUST or REQUIRED level requirements defined in the conformance rules for each section, as indicated by the following format:

192 Rnnnn: Rule text

193 General conformance rules are defined as follows:

194 R1.4-1: To be conformant, the service MUST comply with all the rules defined in this specification. Items marked with
195 MUST are required, and items marked with SHOULD are highly advised to maximize interoperation. Items marked with
196 MAY indicate the preferred implementation for expected features, but interoperation is not affected if they are ignored.

197 R1.4-2: An OVF descriptor MUST NOT use the XML namespace identifier for this specification (see section 1.5) unless it
198 complies with the conformance rules in this specification.

199 4.1 *XML Namespaces*

200 R1.5-1: Conformant services of this specification MUST use this XML namespace Universal
201 Resource Identifier (URI):
202
203 (1) http://schemas.dmtf.org/ovf

204 Table 1 lists XML namespaces used in this specification. The choice of any namespace prefix is arbitrary and not
205 semantically significant.
206

207 **Table 1 – Prefixes and XML Namespaces Used in this Specification**

Prefix	XML Namespace	Specification
ovf	http:// schemas.dmtf.org /ovf/envelope	This specification

208 5 Terms and Definitions

209 For the purposes of this document, the following terms and definitions apply.
210

211 **Appliance**: See Virtual Appliance

212 **CIM**: The Common Information Model standard defined by the DMTF.

213 **Deployment Platform**: The deployment platform is the term for the product that is installing an OVF.

214 **Guest**: The software run when a virtual machine is powered on. It is stored on the virtual disk. The guest is typically an
215 operating system and some user-level applications and services.

216 **Guest Software**: See Guest

217 **Platform**: See Deployment Platform

218 **Virtual Appliance**: A service delivered as a complete software stack installed on one or more virtual machines. A virtual
219 appliance is typically expected to be delivered in an OVF package.

220 **Virtual Hardware**: The hardware seen by the guest software. This includes the CPU, controllers, ethernet devices, and
221 disks.

222 **Virtual Machine**: See VirtualSystem.

223 **VirtualSystem**: A single virtual machine that can be executed. This is a full encapsulation of the virtual hardware, virtual
224 disks, and the meta-data associated with it.

225 **VirtualSystemCollection**: A multi-tier service. A multi-tier service can be a simple set of virtual machines, or it can be a
226 complex service built out of a combination of virtual machines and other VirtualSystemCollections. Since
227 VirtualSystemCollections can be composed, it enables complex multi-tier components.

6 Package Contents

6.1 File Structure

An OVF package shall consist of a single meta-data descriptor and zero or more files. The example below use VMDK disk files, but multiple disk formats are supported.

```
package.ovf
package-disk1.vmdk
package-disk2.vmdk
package.mf          [optional]
```

Optionally, an OVF package can have a manifest file (.mf) containing the SHA-1 digests of each file. The manifest file shall have the same base name as the .ovf file. The format of the .mf file is:

```
<manifest_file> ::= { <file_digest> }
<file_digest>   ::= <algorithm> "(" <file_name> ")= " <digest>
<algorithm>     ::= "SHA1"
<digest>        ::= 160-bit digest in 40 digit hexadecimal
```

For example, a manifest file could contain:

```
SHA1(package.ovf)= 237de026fb285b85528901da058475e56034da95
SHA1(package-disk1.vmdk)= 393a66df214e192ffbfedb78528b5be75cc9e1c3
...
```

An OVF is optionally signed by signing the manifest file. The signed SHA-1 digest is stored in a .cert file along with the base64-encoded X.509 certificate. The .cert file shall have the same base name as the OVF descriptor. The format of the .cert file is:

```
<certificate_file>   ::= <signature_part> <certificate_part>
<signature_part>     ::= <algorithm> "(" <file_name> ")= " <signature>
<algorithm>          ::= "SHA1"
<signature>          ::= 512-bit signature in 128 digit hexadecimal
<certificate_part>   ::= <certificate_header> <certificate_body> <certificate_footer>
<certificate_header> ::= "-----BEGIN CERTIFICATE-----"
<certificate_footer> ::= "-----END CERTIFICATE-----"
<certificate_header> ::= base64-encoded certificate
```

For example, a signed OVF could contain the following files:

```
package.ovf
package-disk1.vmdk
package-disk2.vmdk
package.mf
package.cert
```

An example package.cert file could contain:

```
SHA1(package.mf)= 7f4b8efb8fe20c06df1db68281a63f1b088e19dbf00e5af9db5e8e3e319de
7019db88a3bc699bab6ccd9e09171e21e88ee20b5255cec3fc28350613b2c529089
-----BEGIN CERTIFICATE-----
MIIBgjCCASwCAQQwDQYJKoZIhvcNAQEEBQAwODELMAkGA1UEBhMCQVUxDDAKBgNV
BAgTA1FMRDEbMBkGA1UEAxMSU1NMZWF5L3JzYSB0ZXN0IENBMB4XDTk1MTAwOTIz
MzIwNVoXDTk4MDcwNTIzMzIwNVowYDELMAkGA1UEBhMCQVUxDDAKBgNVBAgTA1FM
RDEZMBcGA1UEChMQTWluY29tIFB0eS4gTHRkLjELMAkGA1UECxMCQ1MxGzAZBgNV
BAMTElNTTGVheSBkZW1vIHNlcnZlcjBcMA0GCSqGSIb3DQEBAQUAA0sAMEgCQQC3
LCXcScWua0PFLkHBLm2VejqpA1F4RQ8q0VjRiPafjx/Z/aWH3ipdMVvuJGa/wFXb
/nDFLDlfWp+oCPwhBtVPAgMBAAEwDQYJKoZIhvcNAQEEBQADQQArNFsihWIjBzb0
DCsU0BvL2bvSwJrPEqFlkDq3F4M6EGutL9axEcANWgbbEdAvNJD1dmEmoWny27Pn
IMs6ZOZB
-----END CERTIFICATE-----
```

6.2 Virtual Disk Formats

OVF does not require any specific disk format be used, but to be compliant with this specification, the disk format shall be identified with a unique format URL that refers to an unencumbered specification on how to interpret the disk format. The specification need not be machine readable, but must be static, and unique so it may be used as a key by software reading an OVF to uniquely determine the format of the disk. The specification must be sufficient for someone skilled in the art to properly interpret the disk format for both reading and writing of disk data.

6.3 Distribution

An OVF package can be stored as a single file using the TAR format. The extension is **.ova** (open virtual appliance or application). For example:

```
D:\virtualappliances\myapp.ova
```

Ordinarily, a TAR extraction tool must scan the whole archive, even if the file requested is found at the front, since replacement files can be appended without modifying the rest of the archive. OVF restricts the TAR format in that duplication is not allowed within the archive, and furthermore requires that the TAR implementation guarantee that the following three files will be placed at the front of the archive, in specified order:

1. `.ovf` descriptor file
2. `.mf` manifest file (optional)
3. `.cert` certificate (optional)

This ensures that it is possible to extract those three files without scanning the entire archive. An OVF TAR archive can still be created using standard TAR packaging tools.

Alternatively, an OVF can be made available as a set of files, for example on a standard web server:

```
http://mywebserver/virtualappliances/myapp.ovf
http://mywebserver/virtualappliances/myapp-disk1.vmdk
http://mywebserver/virtualappliances/myapp-disk2.vmdk
```

7 OVF Descriptor

All meta-data about the appliance and its contents is stored in the OVF descriptor. This is an extensible XML document for encoding information, such as product details, virtual hardware requirements, and licensing.

The XML schema definitions for the OVF descriptor contain the reference on elements and attributes:

- **ovf-envelope.xsd** : Envelope and VM and VM collection content types
- **ovf-section.xsd** : The abstract type for a section
- **ovf-core.xsd** : Core section types (except virtual hardware)
- **ovf-virtualhardware.xsd** : Virtual hardware section types

In the following sections, the semantics, structure, and extensibility framework of the XML descriptor is described in details. These sections are not a replacement for reading the schema definitions, but a complement. Not all details about the set of attributes and elements are present in the semantic description below.

8 OVF Envelope

The envelope is an XML document that describes all meta-data for the virtual machines (including virtual hardware), as well as the structure of the OVF package itself.

The outer-most level consists of four parts:

- A version indication.
- A list of all external files that are part of the OVF package, this is typically virtual disk files and potential ISOs, suspend state files, etc.
- A meta-data part (sections)
- A description of the content, either a single virtual machine (VirtualSystem) or a configuration of multiple virtual machines (VirtualSystemCollection)

The structure is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<ovf:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ovf="http://schemas.dmtf.org/ovf/1/envelope"
    xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cimschema/2/CIM_ VirtualSystemSettingData"
    xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
    ovf:version="0.9">

    <!-- References to all external files -->
    <References>
      <File ovf:id="file1" ovf:href="vmdisk1.vmdk" ovf:size="180114671"/>
      <File ovf:id="file2" ovf:href="vmdisk2.vmdk" ovf:size="4882023564" ovf:chunksize="2147483648"/>
      <File ovf:id="file3" ovf:href="resource.iso" ovf:size="212148764"  ovf:compression="gzip"/>
    </References>

    <!-- Describes meta-information about all virtual disks in the package -->
    <Section xsi:type="ovf:DiskSection_Type">
        <Info>Describes the set of virtual disks</Info>
         ... content of section ...
    </Section>

    <!-- Describes all networks used in the package -->
    <Section xsi:type="ovf:NetworkSection_Type">
         <Info>List of logical networks used in the package</Info>
          ... content of section ...
    </Section>

    <Section xsi:type="ovf:SomeSection_Type" ovf:required="true|false">
        <Info>A plain-text description of the content</Info>
         ... content of section ...
    </Section>

    ... more sections can follow ...

    <Content xsi:type="ovf:VirtualSystem_Type|ovf:VirtualSystemCollection_Type" ovf:id="Some Product">
        ... list of sections ...
    </Content>

</ovf:Envelope>
```

Notes:

- The current version of the specification is 0.9, as indicated with the version attribute of the envelope element.
- The reference part allows a tool to easily determine the integrity of an OVF package without having to parse or interpret (or even understand) the entire structure of the descriptor. Tools can safely manipulate OVFs with no risk of loosing files (e.g. copying or archiving).
- Each file in the reference part may be compressed, using either gzip [RFC1952] or [bzip2]. These may be indicated using `compression="gzip"` or `compression="bzip2"` attributes respectively. Alternatively, if the href is an HTTP or HTTPS URI, then the compression may be specified by the HTTP server, using the HTTP header `Content-Encoding: gzip` or `Content-Encoding: x-bzip2`, see [RFC2616]. Omitting the compression attribute, or specifying it as `"identity"` states that no compression is used.
- Each file in the reference part may be split into chunks to accommodate file size restrictions on certain file systems. Chunking is indicated by the presence of the `chunksize` attribute; this attribute specifies the size of each chunk, except the last, which may be smaller. With `chunksize` specified, the source is not treated as an explicit URI, but as a URI prefix. Each chunk is loaded from `source`.NNNNNNNNN, where `source` is the value given for the `href` attribute, and NNNNNNNNN is the chunk number, starting from 0, and 0-padded to 9 digits. Each chunk is extracted or downloaded separately and assembled on the destination. When chunking is combined with compression, each chunk should be compressed individually, and each chunk should be an equal slice of the uncompressed source, except for the last chunk which may be smaller.
- The section element described in the OVF schema is abstract and cannot be used directly. An `xsi:type` attribute must be specified to define the exact semantics and contents of a section. Similarly for the content element, the `xsi:type` element describes the exact kind of content that is included.
- The `DiskSection` and `NetworkSection` sections are described in detail in the Core Meta-Data Sections.
- The use of `xsi:type` is a core part of making the OVF extensible, since additional type definitions for sections can be added.
- The `required` attribute of a section element specifies whether the information in the section is critical for correct behavior or if it is optional. An OVF application detecting a section element that is optional and that it does not understand is allowed to ignore the section.
- Each section contains a plain-text `Info` field describing the content. This can, for example, be used to warn users when a section is being skipped. The info element supports the `xml:lang` attribute.

There are two content types defined: VirtualSystem and VirtualSystemCollection. The structure of a VirtualSystem is as follows:

```
<Content xsi:type="ovf:VirtualSystem_Type" ovf:id="CoolAppliance">
    <Section xsi:type="...">
        ... content of section ...
    </Section>
    ... more section elements can follow ...
 </Content>
```

A virtual machine content is simply a container of section elements. There are sections for describing virtual hardware, resources, product information, etc. These are described in details later in the specification.

The structure of a VirtualSystemCollection is as follows:

```
<Content xsi:type="ovf:VirtualSystemCollection_Type" ovf:name="Multi-tier Appliance">
   <Section xsi:type="...">
        ... content of section ...
   </Section>
```

```
      ... more section elements can follow ...

      <Content xsi:type="ovf:VirtualSystem_Type|ovf:VirtualSystemCollection_Type" ovf:name="...">
           ... content of element ...
      </Content>
      ... more content elements can follow ...
   </Content>
```

A VirtualSystemCollection is a container of multiple content elements, which themselves can be VirtualSystemCollections. Thus, arbitrary complex configuration can be described. The section elements at the VirtualSystemCollection level, describe product-information, properties, resource requirements, etc.

9 Virtual Hardware Description

The virtual hardware required by a virtual machine is specified in the VirtualHardware section. This specification supports abstract or incomplete hardware descriptions where only the major devices are described. The hypervisor is allowed to create additional virtual hardware controllers and devices, as long as the required devices listed in the descriptor are realized.

This virtual hardware description is based on the CIM classes: *VirtualSystemSettingsData and ResourceAllocationSettingsData*. The XML representation of the CIM model is based on the WS-CIM Mapping (DSP0230). The OVF section for virtual hardware is specified in the ovf-virtualhardware.xsd schema. Please refer to the appendix for the complete listing of the Managed Object Formats (MOFs) and schemas.

The structure is as follows:

```
   <Section xsi:type="VirtualHardwareSection_Type"
     xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
     xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData">
      <Info>500Mb, 1 CPU, 1 disk, 1 nic virtual machine</Info>
      <System>
         ... a VirtualSystemSettingData ...
          <vssd:VirtualSystemType>vmx-4</vssd:VirtualSystemType>
      </System>
      <Item>
           ... a ResourceAllocationSettingData ...
      </Item>
         ... more item elements to follow ...
    </Section>
```

Notes:

- A section describing the virtual hardware has the type: `VirtualHardwareSection_Type`. This is a required section for a VirtualSystem, and disallowed in both the VirtualSystemCollection and envelope elements.

- Multiple VirtualHardwareSection occurrences are allowed. A consumer of the OVF file can select the most appropriate virtual hardware description, typically based on the family attribute. Provided that the guest OS is setup correctly, a single OVF could support multiple hypervisors this way. An alternative approach to supporting multiple hypervisors is to include InstallationSection in the descriptor.

- The optional `VirtualSystemType` element uniquely identifies the family of virtual hardware that is required. Multiple families can be specified with comma separation. For example, a family identifier could be vmx-4 for VMware's 4th generationvirtual hardware or xen-3 for Xen 3 generation virtual hardware.

- The virtual hardware characteristics are described as a series of `Item` elements. This element is an XML representation of the CIM class: *ResourceAllocationSettingData*.

- The CIM ResourceAllocationSettingData can describe all memory and CPU requirements, as well as a description of virtual hardware devices.

- The `required` property on the `Item` specifies whether the realization of the element (for example, a CD-rom or USB controller) is required for correct behavior of the guest software.

- Multiple device subtypes can be specified in an <item> as a comma separated list. For example:

```
<rasd:ResourceSubType>buslogic,lsilogic</rasd:ResourceSubType>
```

The virtual hardware description (`Item` element) is extensible since the CIM-WS Mapping allows additional elements to be included at the end (`xsi:any type`), and additional attributes to be specified (`xsi:AnyAttribute`). New elements will correspond to new properties or subclasses being added to the CIM model.

In the OVF descriptor, an additional boolean `required` attribute is defined. A tool parsing an `Item` element shall act according to the following table:

Child Element	**`required` attribute**	**Action**
Known	true or not specified	MUST interpret `Item`
Known	False	MUST interpret `Item`
Unknown	true or not specified	MUST fail `Item`, unless the `Item` is speficied as not required in which case it can be ignored
Unknown	False	MUST ignore

The general form of the `Item` element is:

```
   <Item xmls:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
schema/2/CIM_ResourceAllocationSettingData">
       <rasd:Caption> A nice description of the content itself </rasd:Caption>
       <rasd:Description> A nice readable description of the kind of content </rasd:Description>
       <rasd:InstanceId> unique key </rasd:InstanceId>
       <rasd:ResourceType> ... </rasd:ResourceType>
       <rasd:OtherResourceType> ... </rasd:OtherResourceType>
       <rasd:ResourceSubType> ... </rasd:ResourceSubType>
       <rasd:HostResource> ... </rasd:HostResource>
       <rasd:AllocationUnit> ... </rasd:AllocationUnit>
       <rasd:VirtualQuantity> ... </rasd:VirtualQuantity>
       <rasd:Reservation> ... </rasd:Reservation>
       <rasd:Limit> ... </rasd:Limit>
       <rasd:Weight> ... </rasd:Weight>
       <rasd:AutomaticAllocation> ... </rasd:AutomaticAllocation>
       <rasd:Parent> ... </rasd:Parent>
       <rasd:Connection> ... </rasd:Connection>
       ... multiple connection elements can be specified ...
       <rasd:Address> ... </rasd:Address>
       <rasd:AddressOnParent> ... </rasd:AddressOnParent>
       <rasd:BusNumber> ... </rasd:BusNumber>
       ... some fields omitted for brevity ...
    </Item>
```

All elements correspond to the *ResourceAllocationSystemSettingsData* CIM class and have the same semantics. The schema is listed in the appendix.

For example, the number of virtual CPUs are described as follows:

```
   <Item xmls:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
schema/2/CIM_ResourceAllocationSettingData">
       <rasd:Caption>2 virtual CPUs, a 300 MHz reservation</rasd:Caption>
       <rasd:Description>The number of virtual CPUs</rasd:Description>
       <rasd:InstanceId>1</rasd:InstanceId>
       <rasd:ResourceType>2</rasd:ResourceType> <!—- 2: Processor --!>
       <rasd:VirtualQuantity>2</rasd:VirtualQuantity>
       <rasd:AllocationUnit>MHz</rasd:AllocationUnit>
       <rasd:Reservation>300</rasd:Reservation>
   </Item>
```

The `Description` element is used to provide additional metadata about the element itself. This enables an application implementing OVF to provide descriptive information about all items, including ones that were unknown at the time the application was written.

Devices such as disks, CD-ROMs, and Network needs a backing from the deployment platform. The requirements on a backing is either specified using the `HostResource` or the `Connection` element.

For an Ethernet Adapter, a logical network name is specified in the `Connection` element. Ethernet Adapters that refer to the same network name within an OVF package MUST be deployed on the same network.

The `HostResource` element is used to either refer to resources included in the OVF descriptor as well as logical devices on the deployment platform. The following syntax is defined:

Type	**Description**
/file/<id>	A reference to a file in the OVF. The id maps to the id on the <File> element.
/disk/<id>	A reference to a virtual disk, as specified in the DiskSection. The id maps to the diskId on the DiskSection element.
/device/<id>	A reference to a device on the deployment platform. The interpretation of this is deployment platform specific and limits portability.

If no backing is specified for a device that requires a backing, the deployment platform shall make an appropriate choice, for example, by prompting the user.

The following table gives a brief overview on how attributes are used to describe virtual devices and controllers:

Attribute	**Usage**
Description	A human-readable description of the meaning of the information. For example, ”Specifies the memory size of the virtual machine”
Caption	A human-readable description of the content. For example, ”256MB memory”
InstanceId	A unique instance id of the element within the section
HostResource	Abstractly specifies how a device shall be connecting to a resource on the deployment platform. Not all devices need a backend. It is specified either as a reference to a file, or a virtual disk included in the OVF package, or a device on the deployment platform (less portable), If not specified, the deployment

	platform shall make an appropriate choice.
resourceType otherResourceType resourceSubtype	Specifies the kind of device that is being described.
AutomaticAllocation	For devices that are connectable, such as floppies, cd-roms, and Ethernet cards, this specifies whether the device should be connected at power on.
Parent	The instanceId of the parent controller (if any)
Connection	For an Ethernet adapter, this specifies the abstract network connection for the virtual machine. All Ethernet adapters that specify the same network connection within an OVF MUST be deployed on the same network.
Address	Device specific. For an Ethernet adapter, this will be the MAC address.
AddressOnParent	For a device, this specifies its location on the controller.
BusNumber	Used for controllers that provides multiple buses (such as IDE and SCSI)

Only fields directly relating to describing devices are mentioned. Refer to the MOF for a complete description of all fields.

10 Core Meta-Data Sections

The following core meta-data sections are defined:

- **DiskSection**: Describes meta-information about all virtual disks in the package.
- **NetworkSection**: Description of logical network(s) used in the package.
- **ResourceAllocationSection**: Specifies reserved, limit, shares on a given resource such as memory or cpu for a VirtualSystemCollection.
- **AnnotationSection**: Specifies a free-form annotation on a virtual machine.
- **ProductSection**: Specifies product-information for an appliance, such as product name and version.
- **PropertySection**: Specifies a set of properties that can be configured for a service or virtual machine.
- **EulaSection**: Specifies a license agreement to be shown during import.
- **StartupSection:** Specifies how a VirtualSystemCollection is powered on.
- **CpuCompatibilitySection**: Specifies any specific CPU compatibility requirements for a virtual machine.
- **OperatingSystemSection**: Specifies the installed guest operating system of a virtual machine.
- **InstallSection**: Specifies that the virtual machine needs to be initially booted to install and configure the software.

Below are the semantics of the core sections along with examples. Please consult the XML schema for a detailed specification of all attributes and elements.

10.1 DiskSection

A DiskSection describes meta-information about all virtual disks in the package. Virtual disks and their meta-data are described outside the virtual hardware to facilitate sharing between virtual machines within an OVF package.

```
<Section xsi:type="DiskSection_Type">
    <Info>Describes the set of virtual disks</Info>
    <Disk ovf:diskId="vmdisk1" ovf:fileRef="file1" ovf:capacity="8589934592"
          ovf:populatedSize="3549324972"
          ovf:format="http://www.vmware.com/specifications/vmdk.html#sparse">
    </Disk>
    <Disk ovf:diskId="vmdisk2" ovf:capacity="536870912"
          ovf:format="http://www.vmware.com/specifications/vmdk.html#sparse">
    </Disk>
    <Disk ovf:diskId="vmdisk3" ovf:capacity="${disk.size}"
          ovf:format="http://www.vmware.com/specifications/vmdk.html#sparse">
    </Disk>
</section>
```

A valid section at the outermost envelope level.

Omitting the fileRef attribute indicates an empty disk. In this case, the disk is created and the entire disk content is zeroed at installation time in accordance with the disk format specified.

Rather than specifying a fixed virtual disk capacity, the capacity for an empty disk can be given using a property, for example `capacity="${disk.size}"`. See section 10.6 for a description of properties.

For non-empty disks, the populated size of the disk can optionally be specified using the `populatedSize` attribute.

For non-empty sparse disks, a parent disk can optionally be specified using the `parentRef` attribute. If a disk block does not exist locally, then lookup occurs in the parent chain.

10.2 NetworkSection

A NetworkSection is used to attach a description to the logical network(s) used in the OVF package.

```
<Section xsi:type="NetworkSection_Type">
    <Info>List of logical networks used in the package</Info>
    <Network ovf:id="red">
        <Caption>Red Network</Caption>
        <Description>The network the Red service will be available on</Description>
    </Network>
</Section>
```

A valid section at the outermost envelope level.

10.3 ResourceAllocationSection

A ResourceAllocationSection is a collection of CIM ResourceAllocationSettingData items and applies only to the VirtualSystemCollection.

```
<Section xsi:type="ResourceAllocationSection_Type"
     xmls:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
```

```
     ovf:required="false">
   <Info>Defines minimum reservations for CPU and memory for the collection of VMs</Info>
   <Item>
      <rasd:Caption>300 MB reservation</rasd:Caption>
      <rasd:InstanceId>0</rasd:InstanceId>
      <rasd:ResourceType>3</rasd:ResourceType>
      <rasd:AllocationUnits>MB</rasd:AllocationUnits>
      <rasd:Reservation>300</rasd:Reservation>
   </Item>
   <Item>
      <rasd:Caption>500 MHz reservation</rasd:Caption>
      <rasd:InstanceId>0</rasd:InstanceId>
      <rasd:ResourceType>4</rasd:ResourceType>
      <rasd:AllocationUnits>MHz</rasd:AllocationUnits>
      <rasd:Reservation>500</rasd:Reservation>
   </Item>
</Section>
```

A valid section for a VirtualSystemCollection entity.

10.4 AnnotationSection

The AnnotationSection is a user-defined annotation on a virtual machine or service.

```
<Section xsi:type="AnnotationSection_Type">
    <Info>An annotation on this service. It can be ignored</Info>
    <Annotation>Contact customer support if you have any problems</Annotation>
</Section>
```

A valid section for a VirtualSystem and a VirtualSystemCollection entity.

10.5 ProductSection

The ProductSection specifies product-information for an appliance, such as product name, version, vendor, etc.

```
<Section xsi:type="ProductSection_Type">
   <Info>Describes product information for the service</Info>
   <Product>SugarCRM Enterprise</Product>
   <Vendor>SugarCRM Corporation</Vendor>
   <Version>4.5</Version>
   <Full-version>4.5-b4523</Full-version>
   <ProductUrl>http://www.sugarcrm.com/</ProductUrl>
   <VendorUrl>http://www.sugarcrm.com</VendorUrl>
   <AppUrl>http://${app.ip}/</AppUrl>
</Section>
```

See PropertySection below for a description of the ${name} property macro syntax.

A valid section for a VirtualSystem and a VirtualSystemCollection entity.

10.6 PropertySection

The PropertySection specifies application-level customization parameters.

This section is particularly relevant to appliances which typically need to be customized during deployment with specific settings such as network identity, and the IP addresses of DNS servers, gateways, and others.

The `transport` attribute on the property section specifies how these parameters are passed to the virtual machine. This supports a pluggable and extensible architecture for providing guest / platform communication protocols. Several protocols can be specified using comma separation.

```
<Section xsi:type="PropertySection_Type" transport="…">
   <Info>Defines the properties used by this service</Info>
   <Property ovf:key="admin.email" ovf:type="string">
```

```
        <Description>Email address of administrator</Description>
    </Property>
    <Property ovf:key="app.ip" ovf:type="string" ovf:defaultValue="192.168.0.2">
          <Description>The IP address of the application server virtual machine</Description>
    </Property>
    <Property ovf:key="db.ip" ovf:type="string" ovf:defaultValue="192.168.0.3">
        <Description>The IP address of the DB server virtual machine</Description>
    </Property>
</Section>
```

Notes:

- A valid section for a VirtualSystem and a VirtualSystemCollection entity.
- The properties specified on a VirtualSystemCollection can also be seen by its immediate children.
- Children can refer to the properties of a parent using macros, such as ${name}.

The following table lists the valid types for properties:

Type	Description
string	A generic string.
string[choice1, choice2, …]	A set of choices.
int	An integer value.
int[x, y, …]	A list of integer choices. Each choice can be specified as a range with format “start-end”.
ip	An IP address in dot-decimal notation.
ip:network	An IP address of the specified network. The network must match a network id specified in the OVF descriptor.

Refer to the OVF Environment section on how properties are propagated to the guest software in virtual machines.

10.7 EulaSection

A EulaSection contains the legal terms for using a particular entity. This license must be shown and accepted during instantiation of an OVF package. Multiple license sections can be present in an OVF. If unattended installs are allowed, all embedded license sections are implicitly accepted.

```
<Section xsi:type="EulaSection_Type" >
    <Info>Licensing agreement</Info>
    <License>
Lorem ipsum dolor sit amet, ligula suspendisse nulla pretium, rhoncus tempor placerat fermentum, enim
integer ad vestibulum volutpat. Nisl rhoncus turpis est, vel elit, congue wisi enim nunc ultricies sit,
magna tincidunt. Maecenas aliquam maecenas ligula nostra, accumsan taciti. Sociis mauris in integer, a
dolor netus non dui aliquet, sagittis felis sodales, dolor sociis mauris, vel eu libero cras. Interdum
at. Eget habitasse elementum est, ipsum purus pede porttitor class, ut adipiscing, aliquet sed auctor,
imperdiet arcu per diam dapibus libero duis. Enim eros in vel, volutpat nec pellentesque leo,
scelerisque.
    </License>
</Section>
```

A valid section for a VirtualSystem and a VirtualSystemCollection entity. This section is also localizable using the `xml:lang` attribute.

10.8 StartupSection

The StartupSection specifies how a VirtualSystemCollection is powered on.

```
<Section xsi:type=”ovf:StartupSection_Type”>
       <item ovf:id=”vm1” ovf:order=”1” ovf:startDelay=”30” ovf:stopDeplay=”20”
             ovf:stopAction=”guestShutdown”/>
       <item ovf:id=”teamA” ovf:order=”2” ovf:startDelay=”30” ovf:stopDeplay=”20”
             ovf:stopAction=”guestShutdown”/>
</Section>
```

Notes:

- An item defaults to order=”1”, startDelay=”0”, stopDelay=”0”, startAction=”powerOn”, stopAction=”powerOff”. Thus, for a trivial startup sequence, no section needs to be specified.
- An item can both be a VirtualSystem or a VirtualSystemCollection. Thus, this element can be invoked recursively.
- The id MUST refer to an entity name within the collection.
- See XML schema in ANNEX B for a list of valid attribute values.

10.9 CpuCompatibilitySection

A CpuCompatibilitySection specifies requirements on the virtualized CPU. This is specified in terms of compatibility with the Intel IA32 CPUID instruction.

```
<Section xsi:type="CpuCompatibilitySection_Type" >
    <Info>CPU Compatibility. This virtual machine requries SSE2</Info>
    <!-- Specifies that it requires the 26th bit in eax on level 1 to be 1.
         All others does not matter -->
    <Level level="1" edx="xxxx:xxTx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx"/>
</Section>
```

Valid section on a VirtualSystem entity

10.10 OperatingSystemSection

A OperatingSystemSection specifies the operating system installed on a virtual machine.

```
<Section xsi:type="OperatingSystemSection_Type" ovf:id="win2003server">
   <Info>Specifies the operating system installed</Info>
   <Description>Windows 2003 Server</Description>
</Section>
```

The valid values for id is defined by the enumeration in CIM_OperatingSystem.

A valid section for a VirtualSystem entity.

10.11 InstallationSection

If this section is present, it indicates that the virtual machine needs to be initially booted to install and configure the software.

```
<Section xsi:type="InstallSection_Type" ovf:transport="…">
   <Info>Specifies that the virtual machine needs to be pre-booted in order to install the software
    </Info>
</Section>
```

The transport attribute specifies a comma separated list of methods for which the guest software supports providing feedback of the installation. See the OVF environment section for further details.

A valid section for a VirtualSystem entity.

11 OVF Environment

The OVF environment defines how the guest software and the deployment platform interacts. This environment allows the guest software to access information about the deployment platform, for example, to access the user specified values for the properties defined in the OVF descriptor. The environment also allows the guest software to provide feedback to the deployment platform, for example, during the software installation phase.

The environment specification is split into a *protocol* part and a *transport* part. The *protocol* defines the format and semantics of the documents and messages that can be exchanged between the guest software and deployment platform. The *transport* defines how the information is communicated between the guest software and deployment platform.

11.1 Environment Document Protocol

The environment document is an extensible XML document that is provided to the guest software about the environment in which it is being executed. How the document is obtained is transport dependent.

The document structure is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<Environment xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:noNamespaceSchemaLocation="http://schemas.dmtf.org/svpc/ovf-environment"
             version="0.9"
              id = "identification of VM from OVF descriptor">
    <!-- Information about hypervisor platform -->
    <Section xsi:type="Platform_Type">
       <kind> Type of virtualization platform </kind>
       <version> Version of virtualization platform </version>
       <vendor> Vendor of virtualization platform </vendor>
       <local> Language and country code </local>
    </Section>

    <!--- Properties defined for this virtual machine -->
    <Section xsi:type="Property_Type">
          <Property key="key" value="value">
          ... more properties ...
    </Section>

    <entity id="id of sibling virtual systems or virtual system containers">
       ... list of sections ...
    </entity>
</Envelope>
```

Notes:

- The *id* attribute in the `Environment` element specifies the id of the virtual machine itself.
- The property section contains the key/value pairs defined for all the properties specified in the OVF descriptor for a particular virtual machine, as well as properties specified for the immediate parent VirturalSystemCollecton, if one exists.
- There is an entity element for each sibling VirtualSystem and VirtualSystemCollection.
- The environment is extensible by providing new section types. A consumer of the document should ignore unknown section types.

11.2 Installation Feedback Protocol

During the installation phase, the following 3 operations are defined:

`success()`: Once the VM has successfully completed its post-install work, it should invoke `success`, to which it will receive an empty response. Once that response is received, the VM should shut down, and once all VMs have shut down, the phase is completed.

`failure(reason)`: If post-installation fails for some reason, the VM should invoke `failure`. This will fail the whole process, including terminating the other VMs in the package. The parameter should be a string that will describe to the user the reason for the failure. Internationalized appliances may use the locale obtained from the OVF environment to guide internationalisation of the error message.

`progress(percentage, message)`: By invoking `progress`, the VM may inform the deployment platform of the progress of the post-install. The percentage argument gives the completion percentage of the phase, as far as that VM is concerned. The message is a string, describing the next task to perform; it may be empty if the appliance wishes to offer no additional information. Internationalized appliances may use the locale obtained from the OVF environment to guide internationalisation of the progress message.

How these operations are invoked is transport dependent.

11.3 Transport

The transport types are specified in the OVF descriptor by the *transport* attribute of the `PropertySection_Type` and `InstallSection_Type`.

OVF does not require any specific transport format to be used, but to be compliant with this specification, the transport format shall be identified with a unique format URL that refers to an unencumbered specification on how to use the transport. The specification need not be machine readable, but must be static, and unique so it may be used as a key by software reading an OVF descriptor to uniquely determine the format. The specification must be sufficient for someone skilled in the art to properly interpret the transport mechanism for implementing the protocols.

ANNEX A: Change Log

(normative)

Version	Date	Author	Description

ANNEX B: OVF XSD

A normative copy of the XML schemas for this specification may be retrieved by resolving the XML namespace URIs for this specification. The current versions of XML schemas are listed below.

Schema: ovf-envelope.xsd

The ovf-envelope.xsd describes overall structure of an OVF descriptor, including the top-level <Envelope> element.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://schemas.dmtf.org/ovf/envelope"
    xmlns:ovf="http://schemas.dmtf.org/ovf/envelope"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <!-- Include virtual hardware schema -->
    <xs:include schemaLocation="ovf-section.xsd"/>
    <xs:include schemaLocation="cim-virtualhardware.xsd"/>
    <xs:include schemaLocation="ovf-core.xsd"/>

    <!-- Root element of a OVF package-->
    <xs:element name="Envelope">
        <xs:complexType>
            <xs:sequence>
                <!--- References to all external files -->
                <xs:element name="References" type="ovf:References_Type"/>

                <!-- Package level meta-data -->
                <xs:element name="Section" type="ovf:Section_Type" minOccurs="0"
                    maxOccurs="unbounded"/>

                <!-- Content. A virtual machine or a vService -->
                <xs:element name="Content" type="ovf:Entity_Type"/>

                <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="signed" type="xs:boolean" use="optional"/>
            <xs:attribute name="manifest" type="xs:boolean" use="optional"/>
            <xs:anyAttribute namespace="##any"/>
        </xs:complexType>
    </xs:element>

    <xs:complexType name="References_Type">
        <xs:sequence>
            <xs:element name="File" type="ovf:File_Type" minOccurs="0"
            maxOccurs="unbounded"/>
            <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:anyAttribute namespace="##any"/>
    </xs:complexType>

    <!--Type for an external reference to a resource -->
    <xs:complexType name="File_Type">
        <xs:sequence>
            <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <!-- Reference key used in other parts of the package -->
```

```
            <xs:attribute name="id" type="xs:string" use="required"/>
            <!-- Same as using a single part element -->
            <xs:attribute name="href" type="xs:string" use="required"/>
            <!-- Size in bytes of the files (if known) -->
            <xs:attribute name="size" type="xs:integer" use="optional"/>
            <!-- Estimated size in bytes of the files (if a good guess is known) -->
            <xs:attribute name="estSize" type="xs:integer" use="optional"/>
            <!-- Compression type (gzip or bzip2) -->
            <xs:attribute name="compression" type="xs:string" use="optional"/>
            <!-- Chunk size (except of last chunk) -->
            <xs:attribute name="chunkSize" type="xs:long" use="optional"/>
            <xs:anyAttribute namespace="##any"/>
        </xs:complexType>

        <!-- Base class for an entity -->
        <xs:complexType name="Entity_Type" abstract="true">
            <xs:sequence>
                <xs:element name="Info" type="ovf:Info_Type" minOccurs="0"
                    maxOccurs="unbounded"/>
                <xs:element name="Section" type="ovf:Section_Type" minOccurs="0"
                maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="id" type="xs:string" use="required"/>
        </xs:complexType>

        <!-- A Virtual Machine Entity -->
        <xs:complexType name="VirtualSystem_Type">
            <xs:complexContent>
                <xs:extension base="ovf:Entity_Type"> </xs:extension>
            </xs:complexContent>
        </xs:complexType>

        <!-- A Composite Service -->
        <xs:complexType name="VirtualSystemCollection_Type">
            <xs:complexContent>
                <xs:extension base="ovf:Entity_Type">
                    <xs:sequence>
                        <xs:element name="Content" type="ovf:Entity_Type"
                        minOccurs="0" maxOccurs="unbounded"/>
                        <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
    maxOccurs="unbounded"/>
                        <xs:any namespace="##other" processContents="lax" minOccurs="0"
    maxOccurs="unbounded"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:schema>
```

Schema: ovf-section.xsd

The schema describes the abstract section element. This is the base type for OVF extensibility and typically included by other schemas, such as the ovf-hardware.xsd.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
    targetNamespace="http://schemas.dmtf.org/ovf/envelope"
    xmlns:ovf="http://schemas.dmtf.org/ovf/envelope"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:import namespace="http://www.w3.org/XML/1998/namespace"
        schemaLocation="http://www.w3.org/2001/xml.xsd"/>

    <!-- The base class for a section. Subclassing this is the most common
          form of extensibility
     -->
    <xs:complexType name="Section_Type" abstract="true">
        <xs:sequence>
            <!-- The info element specifies the meaning of the section. This is
                 typically shown if the section is not understood by the importer
             -->
            <xs:element name="Info" type="ovf:Info_Type" minOccurs="0"
            maxOccurs="unbounded"/>
        </xs:sequence>
        <!-- Whether the import should fail or not, if the section is not
            understood
        -->
        <xs:attribute name="required" type="xs:boolean" use="optional"/>
        <xs:anyAttribute namespace="##any"/>
        <!-- Subtypes defines more specific elements -->
    </xs:complexType>

    <!-- A basic type for a localizable string -->
    <xs:complexType name="Info_Type">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute ref="xml:lang"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:schema>
```

Schema: ovf-core.xsd

The schema describes the non-virtual hardware Section types.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://schemas.dmtf.org/ovf/envelope"
    xmlns:ovf="http://schemas.dmtf.org/ovf/envelope"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:include schemaLocation="ovf-section.xsd"/>
    <xs:import namespace="http://www.w3.org/XML/1998/namespace"
        schemaLocation="http://www.w3.org/2001/xml.xsd"/>

    <!-- A user defined annotation on an entity -->
    <xs:complexType name="AnnotationSection_Type">
        <xs:complexContent>
            <xs:extension base="ovf:Section_Type">
                <xs:sequence>
                    <!-- Several localized annotations can be included -->
                    <xs:element name="Annotation" type="ovf:Info_Type"
                    minOccurs="0" maxOccurs="unbounded"/>
                    <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                    <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:anyAttribute namespace="##any"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <!-- Product information about a virtual appliance -->
    <xs:complexType name="ProductSection_Type">
        <xs:complexContent>
            <xs:extension base="ovf:Section_Type">
                <xs:sequence>
                    <xs:element name="Product" type="ovf:Info_Type"
                        minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="Vendor" type="ovf:Info_Type" minOccurs="0"
                        maxOccurs="unbounded"/>
                    <xs:element name="Version" type="xs:string" minOccurs="0"/>
                    <xs:element name="Full-version" type="xs:string"
                        minOccurs="0"/>
                    <xs:element name="ProductUrl" type="xs:string" minOccurs="0"/>
                    <xs:element name="VendorUrl" type="xs:string" minOccurs="0"/>
                    <xs:element name="AppUrl" type="xs:string" minOccurs="0"/>
                    <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                    <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:anyAttribute namespace="##any"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <!-- Configuration parameters that can be passed to the virtual machine
         for application-level configuration
     -->
    <xs:complexType name="PropertySection_Type">
        <xs:complexContent>
            <xs:extension base="ovf:Section_Type">
                <xs:sequence>
                    <xs:element name="Property" maxOccurs="unbounded">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="Description"
```

```
                                    type="ovf:Info_Type" minOccurs="0"
                                    maxOccurs="unbounded"/>
                                <xs:any namespace="##targetNamespace" processContents="lax"
minOccurs="0" maxOccurs="unbounded"/>
                                <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                            </xs:sequence>
                            <xs:attribute name="key" type="xs:string"/>
                            <xs:attribute name="type" type="xs:string"/>
                            <xs:attribute name="configurableByUser"
                                type="xs:boolean" use="optional"/>
                            <xs:attribute name="configurableAtRuntime"
                                type="xs:boolean" use="optional"/>
                            <xs:attribute name="defaultValue" type="xs:string"
                                use="optional"/>
                            <xs:anyAttribute namespace="##any"/>
                        </xs:complexType>
                    </xs:element>
                    <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                    <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
                <!-- A comma-seperated list of transports that are supported
                     by the virtual machine to access the OVF environment.
                 -->
                <xs:attribute name="transport" type="xs:string" use="optional"/>
                <xs:anyAttribute namespace="##any"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <!-- Provides descriptions for the logical networks used within the
         package. These descriptions are typically used as an aid when the
         package is deployed.
     -->
    <xs:complexType name="NetworkSection_Type">
        <xs:complexContent>
            <xs:extension base="ovf:Section_Type">
                <xs:sequence>
                    <xs:element name="Network" maxOccurs="unbounded">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="Description"
                                    type="ovf:Info_Type" minOccurs="0"
                                    maxOccurs="unbounded"/>
                                <xs:any namespace="##targetNamespace" processContents="lax"
minOccurs="0" maxOccurs="unbounded"/>
                                <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                            </xs:sequence>
                            <xs:attribute name="name" type="xs:string"
                            use="required"/>
                            <xs:anyAttribute namespace="##any"/>
                        </xs:complexType>
                    </xs:element>
                    <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                    <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:anyAttribute namespace="##any"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <!-- Provides meta-information description of the virtual disks in the package -->
    <xs:complexType name="DiskSection_Type">
        <xs:complexContent>
            <xs:extension base="ovf:Section_Type">
```

```
                <xs:sequence>
                    <xs:element name="Disk" type="ovf:VirtualDiskDesc_Type"
                    minOccurs="0" maxOccurs="unbounded"/>
                    <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                    <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:anyAttribute namespace="##any"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>


    <!-- Disk  -->
    <xs:complexType name="VirtualDiskDesc_Type">
        <!-- A logical ID for the virtual disk within this package -->
        <xs:attribute name="diskId" type="xs:string" use="required"/>
        <!-- A file reference to the virtual disk file. If this is not
             specified a blank virtual disk is created of the given size
        -->
        <xs:attribute name="fileRef" type="xs:string" use="optional"/>
        <!-- Capacity in bytes. The capacity can be specified as either
             a size or as a reference to a property using $(property_name)
        -->
        <xs:attribute name="capacity" type="xs:string" use="required"/>
        <!-- Format of the disk. The format is an URL that identifies
             the disk type, e.g., http://www.vmware.com/format/vmdk.html#sparse
        -->
        <xs:attribute name="format" type="xs:string" use="required"/>
        <!-- Populated size of disk. This is an estimation of how much storage
             the disk needs if backed by a non pre-allocated (aka. sparse)
             disk. This size does not take the meta-data into account used
             by a sparse disk.
        -->
        <xs:attribute name="populatedSize" type="xs:long" use="optional"/>
        <!-- Reference to a potential parent disk -->
        <xs:attribute name="parentRef" type="xs:string" use="optional"/>
    </xs:complexType>


    <!-- CPU Architecture requirements for the guest software. -->
    <xs:complexType name="CpuCompatibilitySection_Type">
        <xs:complexContent>
            <xs:extension base="ovf:Section_Type">
                <xs:sequence>
                    <xs:element name="Level" maxOccurs="unbounded">
                        <xs:complexType>
                            <xs:attribute name="level" type="xs:int"
                                use="optional"/>
                            <xs:attribute name="eax" type="xs:string"
                                use="optional"/>
                            <xs:attribute name="ebx" type="xs:string"
                                use="optional"/>
                            <xs:attribute name="ecx" type="xs:string"
                                use="optional"/>
                            <xs:attribute name="edx" type="xs:string"
                                use="optional"/>
                        </xs:complexType>
                    </xs:element>
                    <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                    <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:attribute name="Vendor" type="xs:string"/>
                <xs:anyAttribute namespace="##any"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

```
    <!-- Specification of the operating system installed in the guest -->
    <xs:complexType name="OperatingSystemSection_Type">
        <xs:complexContent>
            <xs:extension base="ovf:Section_Type">
                <xs:sequence>
                    <xs:element name="Description" type="ovf:Info_Type"
                    minOccurs="0" maxOccurs="unbounded"/>
                    <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                    <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
                <!-- The IDs are the enumeration used in
                     CIM_OperatingSystem_Type
                -->
                <xs:attribute name="id" type="xs:string"/>
                <xs:anyAttribute namespace="##any"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <!-- End-User License Agreement -->
    <xs:complexType name="EulaSection_Type">
        <xs:complexContent>
            <xs:extension base="ovf:Section_Type">
                <xs:sequence>
                    <!-- Contains the license agreement in plain text. Several
                        different locales can be specified -->
                    <xs:element name="License" type="ovf:Info_Type"
                    minOccurs="1" maxOccurs="unbounded"/>
                    <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                    <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:anyAttribute namespace="##any"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <!--  For a VirtualSystemCollection, this section is used to specify the order in which the
          contained entities are to be powered on.
    -->
    <xs:complexType name="StartupSection_Type">
        <xs:complexContent>
            <xs:extension base="ovf:Section_Type">
                <xs:sequence>
                    <xs:element name="item" minOccurs="0" maxOccurs="unbounded">
                        <xs:complexType>
                            <!-- Id of entity in collection ->
                            <xs:attribute name="id" type="xs:string"/>
                            <!-- Startup order. Entities are started up starting with lower-numbers
                                 first.  Items with same order identifier may be started up
                                 concurrently or  in any order. The order is reversed for shutdown.
                            -->
                            <xs:attribute name="order" type="xs:int"/>
                            <!-- Delay in seconds to wait for the power on to complete -->
                            <xs:attribute name="startDelay" type="xs:int"/>
                            <!-- Whether to resume power-on sequence,  once the guest reports ok.  -->
                            <xs:attribute name="waitingForGuest" type="xs:boolean"/>
                            <!-- Delay in seconds to wait for the power on to complete -->
                            <xs:attribute name="stopDelay" type="xs:int"/>
                            <!-- Stop action to use. Valid values are: 'powerOn' (default), 'none'. -->
                            <xs:attribute name="startAction" type="xs:string"/>
                            <!-- Stop action to use. Valid values are: 'powerOff' (default),
                                 'guestShutdown', 'suspend'.
                            -->
                            <xs:attribute name="stopAction" type="xs:string"/>
                            <xs:anyAttribute namespace="##any"/>
```

```
                        </xs:complexType>
                    </xs:element>
                    <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                    <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
                <!-- A comma-separated list of transports that the virtual
                    machine supports to provide feedback.
                -->
                <xs:anyAttribute namespace="##any"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <!-- If this section is present, it indicates that the virtual machine
         needs to be initially booted to install and configure the software.
    -->
    <xs:complexType name="InstallSection_Type">
        <xs:complexContent>
            <xs:extension base="ovf:Section_Type">
                <xs:sequence>
                        <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                        <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
                <!-- A comma-separated list of transports that the virtual
                     machine supports to provide feedback.
                -->
                <xs:attribute name="transport" type="xs:string"/>
                <xs:anyAttribute namespace="##any"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:schema>
```

Schema: ovf-virtualhardware.xsd

The schema below describes the XML representation of the virtual hardware.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
    targetNamespace="http://schemas.dmtf.org/ovf/envelope"
    xmlns:ovf="http://schemas.dmtf.org/ovf/envelope"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
    xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData">
    <xs:import namespace="http://www.w3.org/XML/1998/namespace"
        schemaLocation="http://www.w3.org/2001/xml.xsd"/>
    <xs:include schemaLocation="ovf-section.xsd"/>

    <xs:import
        namespace="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
        schemaLocation="cim-vssd.xsd"/>
    <xs:import
        namespace="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
        schemaLocation="cim-rasd.xsd"/>

    <!-- Specifies the virtual hardware for a virtual machine -->
    <xs:complexType name="VirtualHardwareSection_Type">
        <xs:complexContent>
            <xs:extension base="ovf:Section_Type">
                <xs:sequence>
                    <xs:element name="System"
                        type="vssd:CIM_VirtualSystemSettingData_Type"
                        minOccurs="0"/>
                    <xs:element name="Item"
                        type="rasd:CIM_ResourceAllocationSettingData_Type"
                        minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <!-- Specifies a section for resource constraints on a VirtualSystemCollection  -->
    <xs:complexType name="ResourceAllocationSection_Type">
        <xs:complexContent>
            <xs:extension base="ovf:Section_Type">
                <xs:sequence>
                    <xs:element name="Item"
                        type="rasd:CIM_ResourceAllocationSettingData_Type"
                        minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:schema>
```

Schema: cim-rasd.xsd

The schema below describes the XML representation of the CIM ResourceAllocationSettingData .

```
<?xml version='1.0' encoding='utf-8'?>
<xs:schema
    targetNamespace="http://schemas.dmtf.org/wbem/wscim/1/cim-
schema/2/CIM_ResourceAllocationSettingData"
    xmlns:class="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
    xmlns:cim="http://schemas.dmtf.org/wbem/wscim/1/common"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">

    <xs:import namespace="http://schemas.dmtf.org/wbem/wscim/1/common" schemaLocation="cim-
common.xsd"/>

    <xs:element name="Caption" nillable="true" type="cim:cimString"/>

    <xs:element name="Description" nillable="true" type="cim:cimString"/>

    <xs:element name="InstanceId" nillable="true" type="cim:cimString"/>

    <xs:element name="ResourceType" nillable="true">
        <xs:complexType>
            <xs:simpleContent>
                <xs:restriction base="xs:anyType">
                    <xs:simpleType>
                        <xs:union>
                            <xs:simpleType>
                                <xs:restriction base="xs:unsignedShort">
                                    <xs:enumeration value="1"/>   <!-- Other -->
                                    <xs:enumeration value="2"/>   <!-- Computer System  -->
                                    <xs:enumeration value="3"/>    <!-- Processor-->
                                    <xs:enumeration value="4"/>    <!-- Memory-->
                                    <xs:enumeration value="5"/>    <!-- IDE Controller -->
                                    <xs:enumeration value="6"/>    <!-- Parallel SCSI HBA -->
                                    <xs:enumeration value="7"/>    <!-- FC HBA -->
                                    <xs:enumeration value="8"/>    <!-- iSCSI HBA  -->
                                    <xs:enumeration value="9"/>    <!-- IB HCA  -->
                                    <xs:enumeration value="10"/>  <!-- Ethernet Adapter -->
                                    <xs:enumeration value="11"/>  <!-- Other Network Adapter -->
                                    <xs:enumeration value="12"/>  <!-- I/O Slot  -->
                                    <xs:enumeration value="13"/>  <!-- I/O Device -->
                                    <xs:enumeration value="14"/>  <!-- Floppy Drive -->
                                    <xs:enumeration value="15"/>  <!-- CD Drive -->
                                    <xs:enumeration value="16"/>  <!-- DVD drive -->
                                    <xs:enumeration value="17"/>  <!-- Disk Drive -->
                                    <xs:enumeration value="18"/>  <!-- Tape Drive -->
                                    <xs:enumeration value="19"/>  <!-- Storage Extent -->
                                    <xs:enumeration value="20"/>  <!-- Other storage device -->
                                    <xs:enumeration value="21"/>  <!-- Serial port -->
                                    <xs:enumeration value="22"/>  <!-- Parallel port -->
                                    <xs:enumeration value="23"/>  <!-- USB Controller -->
                                    <xs:enumeration value="24"/>  <!-- Graphics controller -->
                                    <xs:enumeration value="25"/>  <!-- IEEE 1394 Controller -->
                                    <xs:enumeration value="26"/>  <!-- Partitionable Unit -->
                                    <xs:enumeration value="27"/>  <!-- Base Partitionable Unit -->
                                    <xs:enumeration value="28"/>  <!-- Power Supply -->
                                    <xs:enumeration value="29"/>  <!-- Cooling Device -->
                                    <xs:enumeration value="30"/>  <!-- PCI Controller -->
                                    <xs:enumeration value="31"/>  <!-- PS2 Controller -->
                                    <xs:enumeration value="32"/>  <!-- SIO Controller -->
                                    <xs:enumeration value="33"/>  <!-- Keyboard -->
                                    <xs:enumeration value="34"/>  <!-- Pointing Device -->
                                </xs:restriction>
                            </xs:simpleType>
                            <xs:simpleType>
```

```
                                <xs:restriction base="xs:unsignedShort">
                                    <xs:minInclusive value="30"/>
                                    <xs:maxInclusive value="32769"/>
                                </xs:restriction>
                            </xs:simpleType>
                            <xs:simpleType>
                                <xs:restriction base="xs:unsignedShort">
                                    <xs:minInclusive value="32768"/>
                                    <xs:maxInclusive value="65535"/>
                                </xs:restriction>
                            </xs:simpleType>
                        </xs:union>
                    </xs:simpleType>
                    <xs:anyAttribute namespace="##any"/>
                </xs:restriction>
            </xs:simpleContent>
        </xs:complexType>
    </xs:element>

    <xs:element name="OtherResourceType" nillable="true" type="cim:cimString"/>

    <xs:element name="ResourceSubType" nillable="true" type="cim:cimString"/>

    <xs:element name="PoolID" nillable="true" type="cim:cimString"/>

    <xs:element name="ConsumerVisibility" nillable="true">
        <xs:complexType>
            <xs:simpleContent>
                <xs:restriction base="xs:anyType">
                    <xs:simpleType>
                        <xs:union>
                            <xs:simpleType>
                                <xs:restriction base="xs:unsignedShort">
                                    <xs:enumeration value="0"/>
                                    <xs:enumeration value="2"/>
                                    <xs:enumeration value="3"/>
                                    <xs:enumeration value="4"/>
                                </xs:restriction>
                            </xs:simpleType>
                            <xs:simpleType>
                                <xs:restriction base="xs:unsignedShort">
                                    <xs:minInclusive value="5"/>
                                    <xs:maxInclusive value="32768"/>
                                </xs:restriction>
                            </xs:simpleType>
                            <xs:simpleType>
                                <xs:restriction base="xs:unsignedShort">
                                    <xs:minInclusive value="32767"/>
                                    <xs:maxInclusive value="65535"/>
                                </xs:restriction>
                            </xs:simpleType>
                        </xs:union>
                    </xs:simpleType>
                    <xs:anyAttribute namespace="##any"/>
                </xs:restriction>
            </xs:simpleContent>
        </xs:complexType>
    </xs:element>

    <xs:element name="HostResource" nillable="true" type="xs:anyType"/>

    <xs:element name="AllocationUnits" nillable="true" type="cim:cimString"/>

    <xs:element name="VirtualQuantity" nillable="true"
        type="cim:cimUnsignedLong"/>

    <xs:element name="Reservation" nillable="true" type="cim:cimUnsignedLong"/>

    <xs:element name="Limit" nillable="true" type="cim:cimUnsignedLong"/>
```

```
1424        <xs:element name="Weight" nillable="true" type="cim:cimUnsignedInt"/>
1425
1426        <xs:element name="AutomaticAllocation" nillable="true" type="cim:cimBoolean"/>
1427
1428        <xs:element name="AutomaticDeallocation" nillable="true"
1429            type="cim:cimBoolean"/>
1430
1431        <xs:element name="Parent" nillable="true" type="cim:cimString"/>
1432
1433        <xs:element name="Connection" nillable="true" type="cim:cimString"/>
1434
1435        <xs:element name="Address" nillable="true" type="cim:cimString"/>
1436
1437        <xs:element name="MappingBehavior" nillable="true">
1438            <xs:complexType>
1439                <xs:simpleContent>
1440                    <xs:restriction base="xs:anyType">
1441                        <xs:simpleType>
1442                            <xs:union>
1443                                <xs:simpleType>
1444                                    <xs:restriction base="xs:unsignedShort">
1445                                        <xs:enumeration value="0"/>
1446                                        <xs:enumeration value="1"/>
1447                                        <xs:enumeration value="2"/>
1448                                        <xs:enumeration value="3"/>
1449                                        <xs:enumeration value="4"/>
1450                                    </xs:restriction>
1451                                </xs:simpleType>
1452                                <xs:simpleType>
1453                                    <xs:restriction base="xs:unsignedShort">
1454                                        <xs:minInclusive value="5"/>
1455                                        <xs:maxInclusive value="32768"/>
1456                                    </xs:restriction>
1457                                </xs:simpleType>
1458                                <xs:simpleType>
1459                                    <xs:restriction base="xs:unsignedShort">
1460                                        <xs:minInclusive value="32767"/>
1461                                        <xs:maxInclusive value="65535"/>
1462                                    </xs:restriction>
1463                                </xs:simpleType>
1464                            </xs:union>
1465                        </xs:simpleType>
1466                        <xs:anyAttribute namespace="##any"/>
1467                    </xs:restriction>
1468                </xs:simpleContent>
1469            </xs:complexType>
1470        </xs:element>
1471
1472        <xs:element name="AddressOnParent" nillable="true" type="cim:cimString"/>
1473
1474        <xs:element name="BusNumber" nillable="true" type="cim:cimUnsignedShort"/>
1475
1476        <xs:complexType name="CIM_ResourceAllocationSettingData_Type">
1477            <xs:sequence>
1478                <xs:element ref="class:Caption" minOccurs="0" maxOccurs="unbounded"/>
1479                <xs:element ref="class:Description" minOccurs="0" maxOccurs="unbounded"/>
1480                <xs:element ref="class:InstanceId" minOccurs="0"/>
1481                <xs:element ref="class:ResourceType" minOccurs="0"/>
1482                <xs:element ref="class:OtherResourceType" minOccurs="0"/>
1483                <xs:element ref="class:ResourceSubType" minOccurs="0"/>
1484                <xs:element ref="class:PoolID" minOccurs="0"/>
1485                <xs:element ref="class:ConsumerVisibility" minOccurs="0"/>
1486                <xs:element ref="class:HostResource" maxOccurs="unbounded" minOccurs="0"/>
1487                <xs:element ref="class:AllocationUnits" minOccurs="0"/>
1488                <xs:element ref="class:VirtualQuantity" minOccurs="0"/>
1489                <xs:element ref="class:Reservation" minOccurs="0"/>
1490                <xs:element ref="class:Limit" minOccurs="0"/>
1491                <xs:element ref="class:Weight" minOccurs="0"/>
1492                <xs:element ref="class:AutomaticAllocation" minOccurs="0"/>
1493                <xs:element ref="class:AutomaticDeallocation" minOccurs="0"/>
```

```
            <xs:element ref="class:Parent" minOccurs="0"/>
            <xs:element ref="class:Connection" maxOccurs="unbounded" minOccurs="0"/>
            <xs:element ref="class:Address" minOccurs="0"/>
            <xs:element ref="class:MappingBehavior" minOccurs="0"/>
            <xs:element ref="class:AddressOnParent" minOccurs="0"/>
            <xs:element ref="class:BusNumber" minOccurs="0"/>
            <xs:any namespace="##other" processContents="lax" minOccurs="0"
                maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:anyAttribute namespace="##any"/>
    </xs:complexType>

    <xs:element name="CIM_ResourceAllocationSettingData"
        type="class:CIM_ResourceAllocationSettingData_Type"/>

</xs:schema>
```

Schema: cim-vssd.xsd

The schema below describes the XML representation of the CIM VirtualSystemSettingData.

```
<?xml version='1.0' encoding='utf-8'?>
<xs:schema
    targetNamespace="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
    xmlns:class="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
    xmlns:cim="http://schemas.dmtf.org/wbem/wscim/1/common"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:import namespace="http://schemas.dmtf.org/wbem/wscim/1/common"
        schemaLocation="cim-common.xsd"/>

    <xs:element name="Caption" nillable="true" type="cim:cimString"/>

    <xs:element name="Description" nillable="true" type="cim:cimString"/>

    <xs:element name="InstanceId" nillable="true" type="cim:cimString"/>

    <xs:element name="VirtualSystemIdentifier" nillable="true" type="cim:cimString"/>

    <xs:element name="VirtualSystemType" nillable="true" type="cim:cimString"/>

    <xs:complexType name="CIM_VirtualSystemSettingData_Type">
        <xs:sequence>
            <xs:element ref="class:Caption" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="class:Description" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="class:InstanceId" minOccurs="0"/>
            <xs:element ref="class:VirtualSystemIdentifier" minOccurs="0"/>
            <xs:element ref="class:VirtualSystemType" minOccurs="0"/>
            <xs:any namespace="##other" processContents="lax" minOccurs="0"
                maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:anyAttribute namespace="##any"/>
    </xs:complexType>

    <xs:element name="CIM_VirtualSystemSettingData"
        type="class:CIM_VirtualSystemSettingData_Type"/>

</xs:schema>
```

Schema: cim-common.xsd

The schema below describes the common CIM types.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema targetNamespace="http://schemas.dmtf.org/wbem/wscim/1/common"
    xmlns:cim="http://schemas.dmtf.org/wbem/wscim/1/common"
    xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">

    <!-- The following are runtime attribute definitions -->
    <xs:attribute name="Key" type="xs:boolean"/>

    <xs:attribute name="Version" type="xs:string"/>

    <!-- The following section defines the extended WS-CIM datatypes -->
    <xs:complexType name="cimDateTime">
        <xs:choice>
            <xs:element name="CIM_DateTime" type="xs:string" nillable="true"/>
            <xs:element name="Interval" type="xs:duration"/>
            <xs:element name="Date" type="xs:date"/>
            <xs:element name="Time" type="xs:time"/>
            <xs:element name="Datetime" type="xs:dateTime"/>
        </xs:choice>
        <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:complexType>

    <xs:complexType name="cimUnsignedByte">
        <xs:simpleContent>
            <xs:extension base="xs:unsignedByte">
                <xs:anyAttribute namespace="##any" processContents="lax"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>


    <xs:complexType name="cimByte">
        <xs:simpleContent>
            <xs:extension base="xs:byte">
                <xs:anyAttribute namespace="##any" processContents="lax"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>

    <xs:complexType name="cimUnsignedShort">
        <xs:simpleContent>
            <xs:extension base="xs:unsignedShort">
                <xs:anyAttribute namespace="##any" processContents="lax"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>

    <xs:complexType name="cimShort">
        <xs:simpleContent>
            <xs:extension base="xs:short">
                <xs:anyAttribute namespace="##any" processContents="lax"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>

    <xs:complexType name="cimUnsignedInt">
        <xs:simpleContent>
            <xs:extension base="xs:unsignedInt">
                <xs:anyAttribute namespace="##any" processContents="lax"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
```

```
1612
1613        <xs:complexType name="cimInt">
1614            <xs:simpleContent>
1615                <xs:extension base="xs:int">
1616                    <xs:anyAttribute namespace="##any" processContents="lax"/>
1617                </xs:extension>
1618            </xs:simpleContent>
1619        </xs:complexType>
1620
1621        <xs:complexType name="cimUnsignedLong">
1622            <xs:simpleContent>
1623                <xs:extension base="xs:unsignedLong">
1624                    <xs:anyAttribute namespace="##any" processContents="lax"/>
1625                </xs:extension>
1626            </xs:simpleContent>
1627        </xs:complexType>
1628
1629
1630        <xs:complexType name="cimLong">
1631            <xs:simpleContent>
1632                <xs:extension base="xs:long">
1633                    <xs:anyAttribute namespace="##any" processContents="lax"/>
1634                </xs:extension>
1635            </xs:simpleContent>
1636        </xs:complexType>
1637
1638        <xs:complexType name="cimString">
1639            <xs:simpleContent>
1640                <xs:extension base="xs:string">
1641                    <xs:anyAttribute namespace="##any" processContents="lax"/>
1642                </xs:extension>
1643            </xs:simpleContent>
1644        </xs:complexType>
1645
1646        <xs:complexType name="cimBoolean">
1647            <xs:simpleContent>
1648                <xs:extension base="xs:boolean">
1649                    <xs:anyAttribute namespace="##any" processContents="lax"/>
1650                </xs:extension>
1651            </xs:simpleContent>
1652        </xs:complexType>
1653
1654        <xs:complexType name="cimFloat">
1655            <xs:simpleContent>
1656                <xs:extension base="xs:float">
1657                    <xs:anyAttribute namespace="##any" processContents="lax"/>
1658                </xs:extension>
1659            </xs:simpleContent>
1660        </xs:complexType>
1661
1662        <xs:complexType name="cimDouble">
1663            <xs:simpleContent>
1664                <xs:extension base="xs:double">
1665                    <xs:anyAttribute namespace="##any" processContents="lax"/>
1666                </xs:extension>
1667            </xs:simpleContent>
1668        </xs:complexType>
1669
1670        <xs:complexType name="cimChar16">
1671            <xs:simpleContent>
1672                <xs:restriction base="cim:cimString">
1673                    <xs:maxLength value="1"/>
1674                    <xs:anyAttribute namespace="##any" processContents="lax"/>
1675                </xs:restriction>
1676            </xs:simpleContent>
1677        </xs:complexType>
1678
1679        <xs:complexType name="cimBase64Binary">
1680            <xs:simpleContent>
1681                <xs:extension base="xs:base64Binary">
```

```
                <xs:anyAttribute namespace="##any" processContents="lax"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>

    <xs:complexType name="cimHexBinary">
        <xs:simpleContent>
            <xs:extension base="xs:hexBinary">
                <xs:anyAttribute namespace="##any" processContents="lax"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>

    <xs:complexType name="cimReference">
        <xs:sequence>
            <xs:any namespace="##other" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:complexType>

    <!-- The following datatypes are used exclusively to define metadata fragments -->
    <xs:attribute name="qualifier" type="xs:boolean"/>

    <xs:complexType name="qualifierString">
        <xs:simpleContent>
            <xs:extension base="cim:cimString">
                <xs:attribute ref="cim:qualifier" use="required"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>

    <xs:complexType name="qualifierBoolean">
        <xs:simpleContent>
            <xs:extension base="cim:cimBoolean">
                <xs:attribute ref="cim:qualifier" use="required"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>

    <xs:complexType name="qualifierUInt32">
        <xs:simpleContent>
            <xs:extension base="cim:cimUnsignedInt">
                <xs:attribute ref="cim:qualifier" use="required"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>

    <xs:complexType name="qualifierSInt64">
        <xs:simpleContent>
            <xs:extension base="cim:cimLong">
                <xs:attribute ref="cim:qualifier" use="required"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>

    <xs:complexType name="qualifierSArray">
        <xs:complexContent>
            <xs:extension base="cim:qualifierString"/>
        </xs:complexContent>
    </xs:complexType>

    <!-- The following element is to be used only for defining metadata -->
    <xs:element name=" DefaultValue" type="xs:anySimpleType"/>
</xs:schema>
```

Schema: ovf-environment.xsd

The schema below describes the XML representation of the OVF environment.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
    targetNamespace="http://schemas.dmtf.org/ovf/environment"
    xmlns:ovfenv="http://schemas.dmtf.org/ovf/environment"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">

    <!-- Root element of a OVF environment-->
    <xs:element name="Environment">
        <xs:complexType>
            <xs:sequence>
                <!-- Entity independent sections -->
                <xs:element name="Section" type="ovfenv:Section_Type" minOccurs="0"
                    maxOccurs="unbounded"/>

                <!-- Information particular to an entity -->
                <xs:element name="Entity" type="ovfenv:Entity_Type" minOccurs="0"
                    maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="version" type="xs:string" use="required"/>
            <xs:attribute name="id" type="xs:string" use="required"/>
        </xs:complexType>
    </xs:element>

    <!-- Abstract type for all sections -->
    <xs:complexType name="Section_Type" abstract="true"/>

    <!-- An entity is a container of sections for a given entity -->
    <xs:complexType name="Entity_Type">
        <xs:sequence>
            <xs:element name="Section" type="ovfenv:Section_Type" minOccurs="0"
                maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="id" type="xs:string" use="required"/>
    </xs:complexType>

    <!-- The key/values of assigned properties for an entity -->
    <xs:complexType name="Property_Type">
        <xs:complexContent>
            <xs:extension base="ovfenv:Section_Type">
                <xs:sequence>
                    <xs:element name="Property"
                        minOccurs="0" maxOccurs="unbounded">
                        <xs:complexType>
                            <xs:attribute name="key" type="xs:string"/>
                            <xs:attribute name="value" type="xs:string"/>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <!-- Information about deployment platform -->
    <xs:complexType name="Platform_Type">
        <xs:complexContent>
            <xs:extension base="ovfenv:Section_Type">
                <xs:sequence>
                    <xs:element name="kind" type="xs:string"/>
                    <xs:element name="version" type="xs:string"/>
                    <xs:element name="vendor" type="xs:string"/>
                    <xs:element name="local" type="xs:string"/>
                </xs:sequence>
```

```
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:schema>
```

ANNEX C: CIM MOFs

MOF: CIM_VirtualSystemSettingData

```
// ===================================================================
// CIM_VirtualSystemSettingData
// ===================================================================
[ Experimental,
  Version("2.14.0"),
  Description(
    "CIM_VirtualSystemSettingData defines the virtual aspects of a "
    "virtual system through a set of virtualization specific "
    "properties. CIM_VirtualSystemSettingData is also used as the "
    "top level class of virtual system configurations. Virtual "
    "system configurations model configuration information about "
    "virtual systems and their components. A virtual system "
    "configuration consists of one top-level instance of class "
    "CIM_VirtualSystemSettingData that aggregates a number of "
    "instances of class CIM_ResourceAllocationSettingData, using "
    "association CIM_ConcreteComponent.\n"
    "Virtual system configurations may for example be used to "
    "reflect configurations of\n"
    "- virtual systems that are defined at a virtualization "
    "platform,\n"
    "- virtual systems that are currently active,\n"
    "- input requests to create new virtual systems,\n"
    "- input requests to modify existing virtual systems, or\n"
    "- snapshots of virtual systems.")]
class CIM_VirtualSystemSettingData : CIM_SettingData {

  [ Description(
      "VirtualSystemIdentifier shall reflect a unique name for "
      "the system as it is used within the virtualization platform. "
      "Note that the VirtualSystemIdentifier is not the hostname "
      "assigned to the operating system instance running within the "
      "virtual system, nor is it an IP address or MAC address "
      "assigned to any of its network ports.\n
      "On create requests VirtualSystemIdentifier may contain"
      "implementation specific rules (like simple patterns or "
      "regular expresssion) that may be interpreted by the "
      "implementation when assigning a VirtualSystemIdentifier."]
   string VirtualSystemIdentifier;

   [ Description(
      "VirtualSystemType shall reflect a particular type of virtual "
      "system. Implementations are commonly capable to support "
      "various implementation defined virtual system types.\n"
      "As stated in the class description, instances of this class "
      "may be used for various purposes. A management application "
      "intending to use an instance of this class as input parameter "
      "to an operation that creates or modifies a virtual system "
      "should first determine the set of valid virtual system types "
      "that are supported by the virtualization platform hosting the "
      "virtual system by inspecting values of array property "
      "VirtualSystemTypesSupported of the instance of class "
      "CIM_VirtualSystemManagementCapabilities that describes the "
      "capabilities of the virtualization platform.")]
     ModelCorrespondence(
      "CIM_VirtualSystemManagementCapabilities.VirtualSystemTypesSupported")]
   string VirtualSystemType;
};
```

MOF: CIM_ResourceAllocationSettingData

The CIM_ResourceAllocationSettingData is currently a Change Request (CR) to CIM. In this specification, we propose a few changes to the CR in order to capture all attributes of virtual hardware. Thise additions are shown in **bold face on grey background**.

```
// Copyright (c) 2007 DMTF.  All rights reserved.
// ==================================================================
// CIM_ResourceAllocationSettingData
// ==================================================================
   [Experimental, Version ("2.16.0"), Description (
      "The ResourceAllocationSettingData class represents settings "
      "specifically related to an allocated resource that are outside "
      "the scope of the CIM class typically used to represent the "
      "resource itself.  These settings include information specific "
      "to the allocation that may not be visible to the consumer "
      "of the resource itself.  For example, a virtual processor may look like "
      "a 2 ghz processor to the consumer (virtual computer system), however "
      "the virtualization system may use time-slicing to schedule the "
      "the virtual processor to only allow it to use 1 ghz. ")] class CIM_ResourceAllocationSettingData
: CIM_SettingData {

      [Description ("The type of resource this allocation setting represents."),
      ValueMap { "1", "2",  "3",  "4",  "5",  "6",  "7",  "8",  "9",
         "10", "11", "12", "13", "14", "15", "16", "17", "18", "19",
         "20", "21", "22", "23", "24", "25", "26", "27", "28", "29",
         "30", "31", "32", "33", "34",
         "..", "0x8000..0xFFFF" },
      Values { "Other", "Computer System", "Processor", "Memory",
         "IDE Controller", "Parallel SCSI HBA", "FC HBA",
         "iSCSI HBA", "IB HCA", "Ethernet Adapter",
         "Other Network Adapter", "I/O Slot", "I/O Device",
         "Floppy Drive", "CD Drive", "DVD drive", "Disk Drive",
         "Tape Drive", "Storage Extent", "Other storage device", "Serial port",
         "Parallel port", "USB Controller", "Graphics controller",
         "IEEE 1394 Controller", "Partitionable Unit",
         "Base Partitionable Unit", "Power Supply", "Cooling Device",
         "PS2 Controller", "SIO Controller", "Keyboard", "Pointing Device",
         "PCI Controller"
         "DMTF reserved", "Vendor Reserved"},
      ModelCorrespondence {
         "CIM_ResourceAllocationSettingData.OtherResourceType",
         "CIM_ResourceAllocationSettingData.ResourceSubType"}]
   uint16 ResourceType;

      [Description ("A string that describes the resource type when a "
         "well defined value is not available and ResourceType has "
         "the value \"Other\"."),
      ModelCorrespondence {
         "CIM_ResourceAllocationSettingData.ResourceType" }]
   string OtherResourceType;

      [Description ("A string describing an implementation specific "
         "sub-type for this resource.  For example, this may "
         "be used to distinguish different models of the same "
         "resource type."),
      ModelCorrespondence {
         "CIM_ResourceAllocationSettingData.ResourceType"}]
   string ResourceSubType;

      [Description ("This property specifies which ResourcePool the "
         "resource is currently allocated from, or which ResourcePool "
         "the resource will be allocated from when the allocation "
         "occurs."),
       ModelCorrespondence { "CIM_ResourcePool.PoolId" }]
   string PoolID;
```

```
1940
1941          [Description (
1942             "Describes the consumers visibility to the allocated  "
1943             "resource.\n"
1944             "A value of \"Passed-Through\" indicates the "
1945             "underlying or host resource is utilized and passed through "
1946             "to the consumer, possibly using partitioning. At least one "
1947             "item shall be present in the HostResource property.  \n"
1948             "A value of \"Virtualized\" indicates the resource is virtualized "
1949             "and may not map directly to an underlying/host resource. "
1950             "Some implementations may support specific assignment "
1951             "for virtualized resources, in which case the host "
1952             "resource(s) are exposed using the HostResource property. \n"
1953             "A value of \"Not represented\" indicates a representation "
1954             "of the resource does not exist within the context of "
1955             "the resource consumer. "),
1956          ValueMap { "0", "2", "3", "4", "..", "32767..65535" },
1957          Values { "Unknown", "Passed-Through", "Virtualized",
1958             "Not represented", "DMTF reserved", "Vendor Reserved" }]
1959       uint16 ConsumerVisibility;
1960
1961          [EmbeddedInstance("CIM_LogicalDevice"), Description (
1962             "This property exposes specific assignment to host or "
1963             "underlying resources.  The embedded instances shall contain "
1964             "only key properties and be treated as Object Paths. If the "
1965             "virtual resource may be scheduled on a number of underlying "
1966             "resources, this property shall be left NULL. In that case, "
1967             "the DeviceAllocatedFromPool or ResourceAllocationFromPool "
1968             "associations may be used to determine the pool of host "
1969             "resources this virtual resource may be scheduled on. "
1970             "If specific assignment is utilized, all underlying "
1971             "resources used by this virtual resource shall be listed "
1972             "in this array.  Typically the array will contain one item, "
1973             "however for aggregate allocations, such as multiple processors, "
1974             "multiple host resources may be specified. ")]
1975       string HostResource[];
1976
1977         [ISPUNIT, Description ("This property specifies the units of allocation "
1978             "used by the Reservation and Limit properties.  For example, "
1979             "when ResourceType=Processor, AllocationUnits may be set to MegaHertz. "
1980             "When ResourceType=Memory, AllocationUnits may be set to MegaBytes"
1981             "The value of this property "
1982             "shall be a legal value of the Programmatic Units qualifier as "
1983             "defined in Appendix C.1 of DSP0004 V2.4 or later.")]
1984       string AllocationUnits;
1985
1986          [Description ( "This property specifies the quantity of resources "
1987             "presented to the consumer.  For example, when "
1988             "ResourceType=Processor,  this property would reflect the "
1989             "number of discrete Processors presented to the virtual "
1990             "computer system.  When ResourceType=Memory, this property "
1991             "could reflect the number of MB reported to the virtual "
1992             "computer system. ")]
1993       uint64 VirtualQuantity;
1994
1995          [Description ("This property specifies the amount of resource "
1996             "guaranteed to be available for this allocation.  On system "
1997             "which support over-commitment of resources, this value "
1998             "is typically used for admission control to prevent an "
1999             "an allocation from being accepted thus preventing starvation.")]
2000       uint64 Reservation;
2001
2002          [Description ("This property specifies the upper bound, or maximum "
2003             "amount of resource that will be granted for this allocation. "
2004             "For example, a system which supports memory paging may support "
2005             "setting the Limit of a Memory allocation below that of the "
2006             "VirtualQuantity, thus forcing paging to occur for this "
2007             "allocation.")]
2008       uint64 Limit;
2009
```

```
      [Description ("This property specifies a relative priority for this "
         "allocation in relation to other allocations from the same "
         "ResourcePool.  This property has no unit of measure, and is "
         "only relevant when compared to other allocations vying for "
         "the same host resources.")]
   uint32 Weight;

      [Description ("This property specifies if the resource will be "
         "automatically allocated.  For example when set to true, "
         "when the consuming virtual computer system is powered on, "
         "this resource would be allocated.  A value of false "
         "indicates the resource must be explicitly allocated.  For "
         "example, the setting may represent removable media "
         "(cdrom, floppy, etc.) where at power on time, the media is"
         "not present.  An explicit operation is required to allocate "
         "the resource.")]
   boolean AutomaticAllocation;

      [Description ("This property specifies if the resource will be "
         "automatically de-allocated.  For example, when set to true, "
         "when the consuming virtual computer system is powered off, "
         "this resource would be de-allocated.  When set to false, the "
         "resource will remain allocated and must be explicitly "
         "de-allocated.")]
   boolean AutomaticDeallocation;

      [Description ("The Parent of the resource."
         "For example, a controller for the current allocation")]
   string Parent;

      [Description ("The thing to which this resource is connected.  "
         "For example, a named network or switch port.")]
   string Connection[];

      [Description ("The address of the resource."
         "For example, the MAC address of a Ethernet port.")]
   string Address;

     [Description ("Specifies how this resource maps to underlying resources"
         "If the HostResource array contains any entries, this property "
         "reflects how the resource maps to those specific resources. "),
      ValueMap { "0", "1", "2", "3", "4", "..", "32767..65535" },
      Values { "Unknown", "Not Supported", "Dedicated",
         "Soft Affinity", "Hard Affinity", "DMTF Reserved", "Vendor Reserved" }]
   uint16 MappingBehavior;

      [Description("Describes the address of this resource in the context "
         "of the Parent.   For example, if the parent is a PCI Controller "
         "this property would specify the PCI slot of the child device.")]
   string AddressOnParent;

      [Description("Specifies the bus on the controller that this describes. "
         " For example, an IDE controller implements multiple buses.")]
   uint16 BusNumber;

};
```

ANNEX C: Authors and Acknowledgements

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

Xen Source

VMware, Inc.

IBM

HP

Dell

Microsoft

Bibliography

This section contains a list of the external references and dependencies for this specification. Dependencies are normative. References are non-normative.

WS-Management
http://www.dmtf.org/standards/published_documents/DSP0226.pdf

WS-CIM Mapping Specification
http://www.dmtf.org/standards/published_documents/DSP0230.pdf

CIM Schema (MOF files)
http://www.dmtf.org/standards/cim/cim_schema_v214

System Virtualization Profile (SVP)
DMTF DSP1042

Virtual System Profile (VSP)
DMTF DSP1057

Resource Allocation Profile (RAP)
DMTF DSP1041
http://www.dmtf.org/standards/published_documents/DSP1041.pdf

Allocation Capabilities Profile (ACP)
DMTF DSP1043

HTTP 1.1
R. Fielding et al, "IETF RFC 2616: Hypertext Transfer Protocol – HTTP/1.1," June 1999.
http://www.ietf.org/rfc/rfc2616.txt

HTTPS
E. Rescorla, "RFC 2818: HTTP over TLS," May 2000.
http://www.ietf.org/rfc/rfc2818.txt

RFC 2119
S. Bradner, "RFC 2119: Key words for use in RFCs to Indicate Requirement Levels," March 1997.
http://www.ietf.org/rfc/rfc2119.txt

XML Schema, Part 1
H. Thompson et al, "XML Schema Part 1: Structures," May 2001.
http://www.w3.org/TR/xmlschema-1/

XML Schema, Part 2
P. Biron et al, "XML Schema Part 2: Datatypes," May 2001.
http://www.w3.org/TR/xmlschema-2/

RFC 3986: Uniform Resource Identifiers (URI): Generic Syntax
http://www.ietf.org/rfc/rfc3986.txt

2117

2118

2119

2120

2121

2122

VMware, Inc. 3401 Hillview Ave., Palo Alto CA, 94304 USA Tel 877-486-9273 Fax 650-427-5001 www.vmware.com