

OVF

Open Virtual Machine Format Specification



version 0.9

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
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66

67

**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**

Copyright © VMware, Inc. and XenSource, Inc. All rights reserved. VMware, the VMware “boxes” logo and design, Virtual SMP and VMotion are registered trademarks or trademarks of VMware, Inc. in the United States and/or other jurisdictions. Xen, XenSource, the “Circle Xen” logo and derivatives thereof are registered trademarks or trademarks of XenSource, Inc. in the United States and other countries. Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation. Linux is a registered trademark of Linus Torvalds. All other marks and names mentioned herein may be trademarks of their respective companies.

No part of this specification (whether in hardcopy or electronic form) may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of VMware, Inc. (VMware), and XenSource, Inc. (XenSource) except as otherwise permitted under copyright law or the terms of that certain Teaming and Non-Disclosure Agreement between VMware and XenSource dated March 23, 2007, as amended from time to time. Please note that the content in this specification is protected under copyright law even if it is not distributed with software that includes an end user license agreement. This specification and the information contained herein is provided on an “AS-IS” basis, is subject to change without notice, and to the maximum extent permitted by applicable law, VMware and XenSource and their respective subsidiaries and affiliates provide the document AS IS AND WITH ALL FAULTS, and hereby disclaim all other warranties and conditions, either express, implied or statutory, including but not limited to, any (if any) implied warranties, duties or conditions of merchantability, of fitness for a particular purpose, of accuracy or completeness of responses, of results, of workmanlike effort, of lack of viruses, and of lack of negligence, all with regard to the document. ALSO, THERE IS NO WARRANTY OR CONDITION OF TITLE, QUIET ENJOYMENT, QUIET POSSESSION, CORRESPONDENCE TO DESCRIPTION OR NON-INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHTS WITH REGARD TO THE DOCUMENT. IN NO EVENT WILL VMWARE, XENSOURCE, OR THEIR SUBSIDIARIES OR AFFILIATES BE LIABLE TO ANY OTHER PARTY FOR THE COST OF PROCURING SUBSTITUTE GOODS OR SERVICES, LOST PROFITS, LOSS OF USE, LOSS OF DATA, OR ANY INCIDENTAL, CONSEQUENTIAL, DIRECT, INDIRECT, OR SPECIAL DAMAGES WHETHER UNDER CONTRACT, TORT, WARRANTY, OR OTHERWISE, ARISING IN ANY WAY OUT OF THIS OR ANY OTHER AGREEMENT RELATING TO THIS DOCUMENT, WHETHER OR NOT SUCH PARTY HAD ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

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**

77

Contents

78			
79	1	Foreword.....	5
80	2	Introduction.....	6
81	3	Scope.....	7
82	3.1	Requirements.....	7
83	3.2	Notations and Terminology.....	7
84	3.3	Notational Conventions.....	7
85	4	Conformance.....	7
86	4.1	XML Namespaces.....	8
87	5	Terms and Definitions.....	8
88	6	Package Contents.....	10
89	6.1	File Structure.....	10
90	6.2	Virtual Disk Formats.....	11
91	6.3	Distribution.....	11
92	7	OVF Descriptor.....	11
93	8	OVF Envelope.....	12
94	9	Virtual Hardware Description.....	14
95	10	Core Meta-Data Sections.....	17
96	10.1	DiskSection.....	18
97	10.2	NetworkSection.....	18
98	10.3	ResourceAllocationSection.....	18
99	10.4	AnnotationSection.....	19
100	10.5	ProductSection.....	19
101	10.6	PropertySection.....	19
102	10.7	EulaSection.....	20
103	10.8	StartupSection.....	21
104	10.9	CpuCompatibilitySection.....	21
105	10.10	OperatingSystemSection.....	21
106	10.11	InstallationSection.....	21
107	11	OVF Environment.....	22
108	11.1	Environment Document Protocol.....	22
109	11.2	Installation Feedback Protocol.....	23
110	11.3	Transport.....	23
111		ANNEX A: Change Log.....	24
112		ANNEX B: OVF XSD.....	25
113		Schema: ovf-envelope.xsd.....	25
114		Schema: ovf-section.xsd.....	27
115		Schema: ovf-core.xsd.....	28
116		Schema: ovf-virtualhardware.xsd.....	33
117		Schema: cim-rasd.xsd.....	34
118		Schema: cim-vssd.xsd.....	38
119		Schema: cim-common.xsd.....	39
120		Schema: ovf-environment.xsd.....	42
121		MOF: CIM_VirtualSystemSettingData.....	44
122		MOF: CIM_ResourceAllocationSettingData.....	45
123		ANNEX C: Authors and Acknowledgements.....	48
124		Bibliography.....	49

125 **1 Foreword**

126 This will be added by the DMTF working group.

127

128 2 Introduction

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

- 131 • **Optimized for distribution**
 132 Supports content verification and integrity checking based on industry standard public key infrastructure, and
 133 provides a basic scheme for management of software licensing.
- 134 • **Optimized for a simple, automated user experience**
 135 Supports validation of the entire package and each virtual machine or meta-data component of the OVF
 136 during the installation phases of the VM lifecycle management process. It also packages with the appliance
 137 relevant user-readable descriptive information that can be use by a virtualization platform to streamline the
 138 installation experience.
- 139 • **Supports both single VM and multi-VM configurations**
 140 Supports both standard single VM packages, and packages containing complex, multi-tier services consisting
 141 of multiple interdependent VMs.
- 142 • **Portable VM packaging**
 143 OVF is virtualization platform neutral, while also enabling platform-specific enhancements to be captured. It
 144 supports the full range of virtual hard disk formats used for VMs today, and is extensible to deal with future
 145 formats that may arise. Virtual machine properties are captured concisely and accurately.
- 146 • **Vendor and platform independent**
 147 The OVF does not rely on the use of a specific host platform, virtualization platform, or guest operating system
 148 (within the appliance).
- 149 • **Extensible**
 150 OVF is immediately useful – and extensible. It is designed to be extended as the industry moves forward with
 151 the virtual appliance technology. It also supports and permits the encoding of vendor specific meta-data to
 152 support specific vertical markets.
- 153 • **Localizable**
 154 Supports user visible descriptions in multiple locales, and supports localization of the interactive processes
 155 during installation of an appliance. This allows a single packaged appliance to serve multiple market
 156 opportunities.
- 157 • **Open standard**
 158 The OVF has arisen from the collaboration of key vendors in the industry, and will be developed in an accepted
 159 industry forum as a future standard for portable virtual machines.

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

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

163 **3 Scope**

164 **3.1 Requirements**

165 This specification intends to meet the following requirements:

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

168 **3.2 Notations and Terminology**

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

170 **3.3 Notational Conventions**

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

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

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

175 Characters are appended to elements and attributes to indicate cardinality:

176 "?" (0 or 1)

177 "*" (0 or more)

178 "+" (1 or more)

179 The character "|" indicates a choice between alternatives.

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

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

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

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

189 **4 Conformance**

190 An implementation is not conformant with this specification if it fails to satisfy one or more of the MUST or REQUIRED
191 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.

277 6.2 Virtual Disk Formats

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

283 6.3 Distribution

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

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

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

- 291 1. `.ovf` descriptor file
- 292 2. `.mf` manifest file (optional)
- 293 3. `.cert` certificate (optional)

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

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

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

300 7 OVF Descriptor

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

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

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

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

311 8 OVF Envelope

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

314 The outer-most level consists of four parts:

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

321 The structure is as follows:

```

322 <?xml version="1.0" encoding="UTF-8"?>
323 <ovf:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
324   xmlns:ovf="http://schemas.dmtf.org/ovf/1/envelope"
325   xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
326   xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
327   ovf:version="0.9">
328
329   <!-- References to all external files -->
330   <References>
331     <File ovf:id="file1" ovf:href="vmdisk1.vmdk" ovf:size="180114671"/>
332     <File ovf:id="file2" ovf:href="vmdisk2.vmdk" ovf:size="4882023564" ovf:chunksize="2147483648"/>
333     <File ovf:id="file3" ovf:href="resource.iso" ovf:size="212148764" ovf:compression="gzip"/>
334   </References>
335
336   <!-- Describes meta-information about all virtual disks in the package -->
337   <Section xsi:type="ovf:DiskSection_Type">
338     <Info>Describes the set of virtual disks</Info>
339     ... content of section ...
340   </Section>
341
342   <!-- Describes all networks used in the package -->
343   <Section xsi:type="ovf:NetworkSection_Type">
344     <Info>List of logical networks used in the package</Info>
345     ... content of section ...
346   </Section>
347
348   <Section xsi:type="ovf:SomeSection_Type" ovf:required="true|false">
349     <Info>A plain-text description of the content</Info>
350     ... content of section ...
351   </Section>
352
353   ... more sections can follow ...
354
355   <Content xsi:type="ovf:VirtualSystem_Type|ovf:VirtualSystemCollection_Type" ovf:id="Some Product">
356     ... list of sections ...
357   </Content>
358
359 </ovf:Envelope>

```

360 Notes:

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

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

```
391 <Content xsi:type="ovf:VirtualSystem_Type" ovf:id="CoolAppliance">
392   <Section xsi:type="...">
393     ... content of section ...
394   </Section>
395   ... more section elements can follow ...
396 </Content>
```

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

399 The structure of a `VirtualSystemCollection` is as follows:

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

```

404 ... more section elements can follow ...
405
406 <Content xsi:type="ovf:VirtualSystem_Type|ovf:VirtualSystemCollection_Type" ovf:name="...">
407   ... content of element ...
408 </Content>
409 ... more content elements can follow ...
410 </Content>

```

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

414 9 Virtual Hardware Description

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

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

423 The structure is as follows:

```

424 <Section xsi:type="VirtualHardwareSection_Type"
425   xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
426   xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData">
427   <Info>500Mb, 1 CPU, 1 disk, 1 nic virtual machine</Info>
428   <System>
429     ... a VirtualSystemSettingData ...
430     <vssd:VirtualSystemType>vmx-4</vssd:VirtualSystemType>
431   </System>
432   <Item>
433     ... a ResourceAllocationSettingData ...
434   </Item>
435   ... more item elements to follow ...
436 </Section>

```

437 Notes:

- 438 • A section describing the virtual hardware has the type: `VirtualHardwareSection_Type`. This is a
 439 required section for a `VirtualSystem`, and disallowed in both the `VirtualSystemCollection` and envelope
 440 elements.
- 441 • Multiple `VirtualHardwareSection` occurrences are allowed. A consumer of the OVF file can select the most
 442 appropriate virtual hardware description, typically based on the family attribute. Provided that the guest OS is
 443 setup correctly, a single OVF could support multiple hypervisors this way. An alternative approach to
 444 supporting multiple hypervisors is to include `InstallationSection` in the descriptor.
- 445 • The optional `VirtualSystemType` element uniquely identifies the family of virtual hardware that is
 446 required. Multiple families can be specified with comma separation. For example, a family identifier could be
 447 `vmx-4` for VMware's 4th generation virtual hardware or `xen-3` for Xen 3 generation virtual hardware.
- 448 • The virtual hardware characteristics are described as a series of `Item` elements. This element is an XML
 449 representation of the CIM class: *ResourceAllocationSettingData*.

- 450 • The CIM ResourceAllocationSettingData can describe all memory and CPU requirements, as well as a
451 description of virtual hardware devices.
- 452 • The `required` property on the `Item` specifies whether the realization of the element (for example, a CD-rom
453 or USB controller) is required for correct behavior of the guest software.
- 454 • Multiple device subtypes can be specified in an `<item>` as a comma separated list. For example:

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

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

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

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

461

462 The general form of the `Item` element is:

```
463 <Item xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-  
464 schema/2/CIM_ResourceAllocationSettingData">  
465 <rasd:Caption> A nice description of the content itself </rasd:Caption>  
466 <rasd:Description> A nice readable description of the kind of content </rasd:Description>  
467 <rasd:InstanceId> unique key </rasd:InstanceId>  
468 <rasd:ResourceType> ... </rasd:ResourceType>  
469 <rasd:OtherResourceType> ... </rasd:OtherResourceType>  
470 <rasd:ResourceSubType> ... </rasd:ResourceSubType>  
471 <rasd:HostResource> ... </rasd:HostResource>  
472 <rasd:AllocationUnit> ... </rasd:AllocationUnit>  
473 <rasd:VirtualQuantity> ... </rasd:VirtualQuantity>  
474 <rasd:Reservation> ... </rasd:Reservation>  
475 <rasd:Limit> ... </rasd:Limit>  
476 <rasd:Weight> ... </rasd:Weight>  
477 <rasd:AutomaticAllocation> ... </rasd:AutomaticAllocation>  
478 <rasd:Parent> ... </rasd:Parent>  
479 <rasd:Connection> ... </rasd:Connection>  
480 ... multiple connection elements can be specified ...  
481 <rasd:Address> ... </rasd:Address>  
482 <rasd:AddressOnParent> ... </rasd:AddressOnParent>  
483 <rasd:BusNumber> ... </rasd:BusNumber>  
484 ... some fields omitted for brevity ...  
485 </Item>
```

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

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

```

489 <Item xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
490 schema/2/CIM_ResourceAllocationSettingData">
491 <rasd:Caption>2 virtual CPUs, a 300 MHz reservation</rasd:Caption>
492 <rasd:Description>The number of virtual CPUs</rasd:Description>
493 <rasd:InstanceId>1</rasd:InstanceId>
494 <rasd:ResourceType>2</rasd:ResourceType> <!-- 2: Processor --!>
495 <rasd:VirtualQuantity>2</rasd:VirtualQuantity>
496 <rasd:AllocationUnit>MHz</rasd:AllocationUnit>
497 <rasd:Reservation>300</rasd:Reservation>
498 </Item>
499

```

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

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

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

507 The `HostResource` element is used to either refer to resources included in the OVF descriptor as well as logical
 508 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.

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

511
 512 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 instancelid 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)

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

515 10 Core Meta-Data Sections

516 The following core meta-data sections are defined:

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

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

532 10.1 DiskSection

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

```
535
536 <Section xsi:type="DiskSection_Type">
537   <Info>Describes the set of virtual disks</Info>
538   <Disk ovf:diskId="vmdisk1" ovf:fileRef="file1" ovf:capacity="8589934592"
539     ovf:populatedSize="3549324972"
540     ovf:format="http://www.vmware.com/specifications/vmdk.html#sparse">
541   </Disk>
542   <Disk ovf:diskId="vmdisk2" ovf:capacity="536870912"
543     ovf:format="http://www.vmware.com/specifications/vmdk.html#sparse">
544   </Disk>
545   <Disk ovf:diskId="vmdisk3" ovf:capacity="{disk.size}"
546     ovf:format="http://www.vmware.com/specifications/vmdk.html#sparse">
547   </Disk>
548 </Section>
```

549 A valid section at the outermost envelope level.

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

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

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

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

557 10.2 NetworkSection

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

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

566 A valid section at the outermost envelope level.

567 10.3 ResourceAllocationSection

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

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

```

572     ovf:required="false">
573     <Info>Defines minimum reservations for CPU and memory for the collection of VMs</Info>
574     <Item>
575         <rasd:Caption>300 MB reservation</rasd:Caption>
576         <rasd:InstanceId>0</rasd:InstanceId>
577         <rasd:ResourceType>3</rasd:ResourceType>
578         <rasd:AllocationUnits>MB</rasd:AllocationUnits>
579         <rasd:Reservation>300</rasd:Reservation>
580     </Item>
581     <Item>
582         <rasd:Caption>500 MHz reservation</rasd:Caption>
583         <rasd:InstanceId>0</rasd:InstanceId>
584         <rasd:ResourceType>4</rasd:ResourceType>
585         <rasd:AllocationUnits>MHz</rasd:AllocationUnits>
586         <rasd:Reservation>500</rasd:Reservation>
587     </Item>
588 </Section>

```

589 A valid section for a VirtualSystemCollection entity.

590 10.4 AnnotationSection

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

```

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

```

596 A valid section for a VirtualSystem and a VirtualSystemCollection entity.

597 10.5 ProductSection

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

```

599 <Section xsi:type="ProductSection_Type">
600     <Info>Describes product information for the service</Info>
601     <Product>SugarCRM Enterprise</Product>
602     <Vendor>SugarCRM Corporation</Vendor>
603     <Version>4.5</Version>
604     <Full-version>4.5-b4523</Full-version>
605     <ProductUrl>http://www.sugarcrm.com/</ProductUrl>
606     <VendorUrl>http://www.sugarcrm.com</VendorUrl>
607     <AppUrl>http://{app.ip}/</AppUrl>
608 </Section>

```

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

610 A valid section for a VirtualSystem and a VirtualSystemCollection entity.

611 10.6 PropertySection

612 The PropertySection specifies application-level customization parameters.

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

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

```

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

```

```

621     <Description>Email address of administrator</Description>
622   </Property>
623   <Property ovf:key="app.ip" ovf:type="string" ovf:defaultValue="192.168.0.2">
624     <Description>The IP address of the application server virtual machine</Description>
625   </Property>
626   <Property ovf:key="db.ip" ovf:type="string" ovf:defaultValue="192.168.0.3">
627     <Description>The IP address of the DB server virtual machine</Description>
628   </Property>
629 </Section>

```

630 Notes:

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

634 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.

635

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

637 10.7 EulaSection

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

```

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

```

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

655 10.8 StartupSection

656 The StartupSection specifies how a VirtualSystemCollection is powered on.

```
657 <Section xsi:type="ovf:StartupSection_Type">
658     <item ovf:id="vm1" ovf:order="1" ovf:startDelay="30" ovf:stopDeploy="20"
659         ovf:stopAction="guestShutdown"/>
660     <item ovf:id="teamA" ovf:order="2" ovf:startDelay="30" ovf:stopDeploy="20"
661         ovf:stopAction="guestShutdown"/>
662 </Section>
```

663

664 Notes:

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

670

671 10.9 CpuCompatibilitySection

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

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

680 Valid section on a VirtualSystem entity

681 10.10 OperatingSystemSection

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

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

687

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

689 A valid section for a VirtualSystem entity.

690 10.11 InstallationSection

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

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

697

698 The transport attribute specifies a comma separated list of methods for which the guest software supports providing
 699 feedback of the installation. See the OVF environment section for further details.
 700 A valid section for a VirtualSystem entity.

701 11 OVF Environment

702
 703 The OVF environment defines how the guest software and the deployment platform interacts. This environment allows
 704 the guest software to access information about the deployment platform, for example, to access the user specified
 705 values for the properties defined in the OVF descriptor. The environment also allows the guest software to provide
 706 feedback to the deployment platform, for example, during the software installation phase.

707
 708 The environment specification is split into a *protocol* part and a *transport* part. The *protocol* defines the format and
 709 semantics of the documents and messages that can be exchanged between the guest software and deployment
 710 platform. The *transport* defines how the information is communicated between the guest software and deployment
 711 platform.

712 11.1 Environment Document Protocol

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

715
 716 The document structure is as follows:

```

717
718 <?xml version="1.0" encoding="UTF-8"?>
719 <Environment xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
720     xsi:noNamespaceSchemaLocation="http://schemas.dmtf.org/svpc/ovf-environment"
721     version="0.9"
722     id = "identification of VM from OVF descriptor">
723   <!-- Information about hypervisor platform -->
724   <Section xsi:type="Platform_Type">
725     <kind> Type of virtualization platform </kind>
726     <version> Version of virtualization platform </version>
727     <vendor> Vendor of virtualization platform </vendor>
728     <local> Language and country code </local>
729   </Section>
730
731   <!-- Properties defined for this virtual machine -->
732   <Section xsi:type="Property_Type">
733     <Property key="key" value="value">
734       ... more properties ...
735   </Section>
736
737   <entity id="id of sibling virtual systems or virtual system containers">
738     ... list of sections ...
739   </entity>
740 </Envelope>
  
```

741
 742 Notes:

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

750 11.2 Installation Feedback Protocol

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

752

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

756

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

761

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

767

768 How these operations are invoked is transport dependent.

769 11.3 Transport

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

772

773 OVF does not require any specific transport format to be used, but to be compliant with this specification, the transport
774 format shall be identified with a unique format URL that refers to an unencumbered specification on how to use the
775 transport. The specification need not be machine readable, but must be static, and unique so it may be used as a key
776 by software reading an OVF descriptor to uniquely determine the format. The specification must be sufficient for
777 someone skilled in the art to properly interpret the transport mechanism for implementing the protocols.
778

781 **ANNEX B: OVF XSD**

782 A normative copy of the XML schemas for this specification may be retrieved by resolving the XML namespace URIs for
 783 this specification. The current versions of XML schemas are listed below.
 784

785 **Schema: ovf-envelope.xsd**

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

```

787 <?xml version="1.0" encoding="UTF-8"?>
788 <xs:schema targetNamespace="http://schemas.dmtf.org/ovf/envelope"
789   xmlns:ovf="http://schemas.dmtf.org/ovf/envelope"
790   xmlns:xs="http://www.w3.org/2001/XMLSchema">
791   <!-- Include virtual hardware schema -->
792   <xs:include schemaLocation="ovf-section.xsd"/>
793   <xs:include schemaLocation="cim-virtualhardware.xsd"/>
794   <xs:include schemaLocation="ovf-core.xsd"/>
795
796   <!-- Root element of a OVF package-->
797   <xs:element name="Envelope">
798     <xs:complexType>
799       <xs:sequence>
800         <!-- References to all external files -->
801         <xs:element name="References" type="ovf:References_Type"/>
802
803         <!-- Package level meta-data -->
804         <xs:element name="Section" type="ovf:Section_Type" minOccurs="0"
805           maxOccurs="unbounded"/>
806
807         <!-- Content. A virtual machine or a vService -->
808         <xs:element name="Content" type="ovf:Entity_Type"/>
809
810         <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
811 maxOccurs="unbounded"/>
812         <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
813       </xs:sequence>
814       <xs:attribute name="signed" type="xs:boolean" use="optional"/>
815       <xs:attribute name="manifest" type="xs:boolean" use="optional"/>
816       <xs:anyAttribute namespace="##any"/>
817     </xs:complexType>
818   </xs:element>
819
820   <xs:complexType name="References_Type">
821     <xs:sequence>
822       <xs:element name="File" type="ovf:File_Type" minOccurs="0"
823 maxOccurs="unbounded"/>
824       <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
825 maxOccurs="unbounded"/>
826       <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
827     </xs:sequence>
828     <xs:anyAttribute namespace="##any"/>
829   </xs:complexType>
830
831   <!--Type for an external reference to a resource -->
832   <xs:complexType name="File_Type">
833     <xs:sequence>
834       <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
835 maxOccurs="unbounded"/>
836       <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
837     </xs:sequence>
838   </xs:complexType>
839   <!-- Reference key used in other parts of the package -->

```

```

839     <xs:attribute name="id" type="xs:string" use="required"/>
840     <!-- Same as using a single part element -->
841     <xs:attribute name="href" type="xs:string" use="required"/>
842     <!-- Size in bytes of the files (if known) -->
843     <xs:attribute name="size" type="xs:integer" use="optional"/>
844     <!-- Estimated size in bytes of the files (if a good guess is known) -->
845     <xs:attribute name="estSize" type="xs:integer" use="optional"/>
846     <!-- Compression type (gzip or bzip2) -->
847     <xs:attribute name="compression" type="xs:string" use="optional"/>
848     <!-- Chunk size (except of last chunk) -->
849     <xs:attribute name="chunkSize" type="xs:long" use="optional"/>
850     <xs:anyAttribute namespace="##any"/>
851 </xs:complexType>
852
853 <!-- Base class for an entity -->
854 <xs:complexType name="Entity_Type" abstract="true">
855   <xs:sequence>
856     <xs:element name="Info" type="ovf:Info_Type" minOccurs="0"
857       maxOccurs="unbounded"/>
858     <xs:element name="Section" type="ovf:Section_Type" minOccurs="0"
859       maxOccurs="unbounded"/>
860   </xs:sequence>
861   <xs:attribute name="id" type="xs:string" use="required"/>
862 </xs:complexType>
863
864 <!-- A Virtual Machine Entity -->
865 <xs:complexType name="VirtualSystem_Type">
866   <xs:complexContent>
867     <xs:extension base="ovf:Entity_Type"> </xs:extension>
868   </xs:complexContent>
869 </xs:complexType>
870
871 <!-- A Composite Service -->
872 <xs:complexType name="VirtualSystemCollection_Type">
873   <xs:complexContent>
874     <xs:extension base="ovf:Entity_Type">
875       <xs:sequence>
876         <xs:element name="Content" type="ovf:Entity_Type"
877           minOccurs="0" maxOccurs="unbounded"/>
878         <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
879           maxOccurs="unbounded"/>
880         <xs:any namespace="##other" processContents="lax" minOccurs="0"
881           maxOccurs="unbounded"/>
882       </xs:sequence>
883     </xs:extension>
884   </xs:complexContent>
885 </xs:complexType>
886 </xs:schema>
887

```

888 **Schema: ovf-section.xsd**

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

```

891 <?xml version="1.0" encoding="UTF-8"?>
892 <xs:schema
893     targetNamespace="http://schemas.dmtf.org/ovf/envelope"
894     xmlns:ovf="http://schemas.dmtf.org/ovf/envelope"
895     xmlns:xs="http://www.w3.org/2001/XMLSchema">
896     <xs:import namespace="http://www.w3.org/XML/1998/namespace"
897         schemaLocation="http://www.w3.org/2001/xml.xsd"/>
898
899     <!-- The base class for a section. Subclassing this is the most common
900         form of extensibility
901     -->
902     <xs:complexType name="Section_Type" abstract="true">
903         <xs:sequence>
904             <!-- The info element specifies the meaning of the section. This is
905                 typically shown if the section is not understood by the importer
906             -->
907             <xs:element name="Info" type="ovf:Info_Type" minOccurs="0"
908                 maxOccurs="unbounded"/>
909         </xs:sequence>
910         <!-- Whether the import should fail or not, if the section is not
911             understood
912         -->
913         <xs:attribute name="required" type="xs:boolean" use="optional"/>
914         <xs:anyAttribute namespace="##any"/>
915         <!-- Subtypes defines more specific elements -->
916     </xs:complexType>
917
918     <!-- A basic type for a localizable string -->
919     <xs:complexType name="Info_Type">
920         <xs:simpleContent>
921             <xs:extension base="xs:string">
922                 <xs:attribute ref="xml:lang"/>
923             </xs:extension>
924         </xs:simpleContent>
925     </xs:complexType>
926 </xs:schema>
927

```

928 **Schema: ovf-core.xsd**

929 The schema describes the non-virtual hardware Section types.

```

930 <?xml version="1.0" encoding="UTF-8"?>
931 <xs:schema targetNamespace="http://schemas.dmtf.org/ovf/envelope"
932   xmlns:ovf="http://schemas.dmtf.org/ovf/envelope"
933   xmlns:xs="http://www.w3.org/2001/XMLSchema">
934   <xs:include schemaLocation="ovf-section.xsd"/>
935   <xs:import namespace="http://www.w3.org/XML/1998/namespace"
936     schemaLocation="http://www.w3.org/2001/xml.xsd"/>
937
938   <!-- A user defined annotation on an entity -->
939   <xs:complexType name="AnnotationSection_Type">
940     <xs:complexContent>
941       <xs:extension base="ovf:Section_Type">
942         <xs:sequence>
943           <!-- Several localized annotations can be included -->
944           <xs:element name="Annotation" type="ovf:Info_Type"
945             minOccurs="0" maxOccurs="unbounded"/>
946           <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
947 maxOccurs="unbounded"/>
948           <xs:any namespace="##other" processContents="lax" minOccurs="0"
949 maxOccurs="unbounded"/>
950         </xs:sequence>
951         <xs:anyAttribute namespace="##any"/>
952       </xs:extension>
953     </xs:complexContent>
954   </xs:complexType>
955
956   <!-- Product information about a virtual appliance -->
957   <xs:complexType name="ProductSection_Type">
958     <xs:complexContent>
959       <xs:extension base="ovf:Section_Type">
960         <xs:sequence>
961           <xs:element name="Product" type="ovf:Info_Type"
962             minOccurs="0" maxOccurs="unbounded"/>
963           <xs:element name="Vendor" type="ovf:Info_Type" minOccurs="0"
964             maxOccurs="unbounded"/>
965           <xs:element name="Version" type="xs:string" minOccurs="0"/>
966           <xs:element name="Full-version" type="xs:string"
967             minOccurs="0"/>
968           <xs:element name="ProductUrl" type="xs:string" minOccurs="0"/>
969           <xs:element name="VendorUrl" type="xs:string" minOccurs="0"/>
970           <xs:element name="AppUrl" type="xs:string" minOccurs="0"/>
971           <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
972 maxOccurs="unbounded"/>
973           <xs:any namespace="##other" processContents="lax" minOccurs="0"
974 maxOccurs="unbounded"/>
975         </xs:sequence>
976         <xs:anyAttribute namespace="##any"/>
977       </xs:extension>
978     </xs:complexContent>
979   </xs:complexType>
980
981   <!-- Configuration parameters that can be passed to the virtual machine
982     for application-level configuration
983   -->
984   <xs:complexType name="PropertySection_Type">
985     <xs:complexContent>
986       <xs:extension base="ovf:Section_Type">
987         <xs:sequence>
988           <xs:element name="Property" maxOccurs="unbounded">
989             <xs:complexType>
990               <xs:sequence>
991                 <xs:element name="Description"

```

```

992                                     type="ovf:Info_Type" minOccurs="0"
993                                     maxOccurs="unbounded"/>
994                                     <xs:any namespace="##targetNamespace" processContents="lax"
995 minOccurs="0" maxOccurs="unbounded"/>
996                                     <xs:any namespace="##other" processContents="lax" minOccurs="0"
997 maxOccurs="unbounded"/>
998                                 </xs:sequence>
999                                 <xs:attribute name="key" type="xs:string"/>
1000                                <xs:attribute name="type" type="xs:string"/>
1001                                <xs:attribute name="configurableByUser"
1002                                     type="xs:boolean" use="optional"/>
1003                                <xs:attribute name="configurableAtRuntime"
1004                                     type="xs:boolean" use="optional"/>
1005                                <xs:attribute name="defaultValue" type="xs:string"
1006                                     use="optional"/>
1007                                <xs:anyAttribute namespace="##any"/>
1008                            </xs:complexType>
1009                        </xs:element>
1010                        <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
1011 maxOccurs="unbounded"/>
1012                        <xs:any namespace="##other" processContents="lax" minOccurs="0"
1013 maxOccurs="unbounded"/>
1014                    </xs:sequence>
1015                    <!-- A comma-separated list of transports that are supported
1016                         by the virtual machine to access the OVF environment.
1017                    -->
1018                    <xs:attribute name="transport" type="xs:string" use="optional"/>
1019                    <xs:anyAttribute namespace="##any"/>
1020                </xs:extension>
1021            </xs:complexContent>
1022        </xs:complexType>
1023
1024        <!-- Provides descriptions for the logical networks used within the
1025             package. These descriptions are typically used as an aid when the
1026             package is deployed.
1027        -->
1028        <xs:complexType name="NetworkSection_Type">
1029            <xs:complexContent>
1030                <xs:extension base="ovf:Section_Type">
1031                    <xs:sequence>
1032                        <xs:element name="Network" maxOccurs="unbounded">
1033                            <xs:complexType>
1034                                <xs:sequence>
1035                                    <xs:element name="Description"
1036                                         type="ovf:Info_Type" minOccurs="0"
1037                                         maxOccurs="unbounded"/>
1038                                    <xs:any namespace="##targetNamespace" processContents="lax"
1039 minOccurs="0" maxOccurs="unbounded"/>
1040                                    <xs:any namespace="##other" processContents="lax" minOccurs="0"
1041 maxOccurs="unbounded"/>
1042                                </xs:sequence>
1043                                <xs:attribute name="name" type="xs:string"
1044                                     use="required"/>
1045                                <xs:anyAttribute namespace="##any"/>
1046                            </xs:complexType>
1047                        </xs:element>
1048                        <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
1049 maxOccurs="unbounded"/>
1050                        <xs:any namespace="##other" processContents="lax" minOccurs="0"
1051 maxOccurs="unbounded"/>
1052                    </xs:sequence>
1053                    <xs:anyAttribute namespace="##any"/>
1054                </xs:extension>
1055            </xs:complexContent>
1056        </xs:complexType>
1057
1058        <!-- Provides meta-information description of the virtual disks in the package -->
1059        <xs:complexType name="DiskSection_Type">
1060            <xs:complexContent>
1061                <xs:extension base="ovf:Section_Type">

```

```

1062         <xs:sequence>
1063             <xs:element name="Disk" type="ovf:VirtualDiskDesc_Type"
1064                 minOccurs="0" maxOccurs="unbounded" />
1065             <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
1066 maxOccurs="unbounded" />
1067             <xs:any namespace="##other" processContents="lax" minOccurs="0"
1068 maxOccurs="unbounded" />
1069         </xs:sequence>
1070         <xs:anyAttribute namespace="##any" />
1071     </xs:extension>
1072 </xs:complexContent>
1073 </xs:complexType>
1074
1075
1076 <!-- Disk -->
1077 <xs:complexType name="VirtualDiskDesc_Type">
1078     <!-- A logical ID for the virtual disk within this package -->
1079     <xs:attribute name="diskId" type="xs:string" use="required" />
1080     <!-- A file reference to the virtual disk file. If this is not
1081         specified a blank virtual disk is created of the given size
1082     -->
1083     <xs:attribute name="fileRef" type="xs:string" use="optional" />
1084     <!-- Capacity in bytes. The capacity can be specified as either
1085         a size or as a reference to a property using $(property_name)
1086     -->
1087     <xs:attribute name="capacity" type="xs:string" use="required" />
1088     <!-- Format of the disk. The format is an URL that identifies
1089         the disk type, e.g., http://www.vmware.com/format/vmdk.html#sparse
1090     -->
1091     <xs:attribute name="format" type="xs:string" use="required" />
1092     <!-- Populated size of disk. This is an estimation of how much storage
1093         the disk needs if backed by a non pre-allocated (aka. sparse)
1094         disk. This size does not take the meta-data into account used
1095         by a sparse disk.
1096     -->
1097     <xs:attribute name="populatedSize" type="xs:long" use="optional" />
1098     <!-- Reference to a potential parent disk -->
1099     <xs:attribute name="parentRef" type="xs:string" use="optional" />
1100 </xs:complexType>
1101
1102
1103 <!-- CPU Architecture requirements for the guest software. -->
1104 <xs:complexType name="CpuCompatibilitySection_Type">
1105     <xs:complexContent>
1106         <xs:extension base="ovf:Section_Type">
1107             <xs:sequence>
1108                 <xs:element name="Level" maxOccurs="unbounded">
1109                     <xs:complexType>
1110                         <xs:attribute name="level" type="xs:int"
1111                             use="optional" />
1112                         <xs:attribute name="eax" type="xs:string"
1113                             use="optional" />
1114                         <xs:attribute name="ebx" type="xs:string"
1115                             use="optional" />
1116                         <xs:attribute name="ecx" type="xs:string"
1117                             use="optional" />
1118                         <xs:attribute name="edx" type="xs:string"
1119                             use="optional" />
1120                     </xs:complexType>
1121                 </xs:element>
1122                 <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
1123 maxOccurs="unbounded" />
1124                 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1125 maxOccurs="unbounded" />
1126             </xs:sequence>
1127             <xs:attribute name="Vendor" type="xs:string" />
1128             <xs:anyAttribute namespace="##any" />
1129         </xs:extension>
1130     </xs:complexContent>
1131 </xs:complexType>

```

```

1132
1133 <!-- Specification of the operating system installed in the guest -->
1134 <xs:complexType name="OperatingSystemSection_Type">
1135   <xs:complexContent>
1136     <xs:extension base="ovf:Section_Type">
1137       <xs:sequence>
1138         <xs:element name="Description" type="ovf:Info_Type"
1139           minOccurs="0" maxOccurs="unbounded"/>
1140         <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
1141 maxOccurs="unbounded"/>
1142         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1143 maxOccurs="unbounded"/>
1144       </xs:sequence>
1145       <!-- The IDs are the enumeration used in
1146         CIM_OperatingSystem_Type
1147       -->
1148       <xs:attribute name="id" type="xs:string"/>
1149       <xs:anyAttribute namespace="##any"/>
1150     </xs:extension>
1151   </xs:complexContent>
1152 </xs:complexType>
1153
1154 <!-- End-User License Agreement -->
1155 <xs:complexType name="EulaSection_Type">
1156   <xs:complexContent>
1157     <xs:extension base="ovf:Section_Type">
1158       <xs:sequence>
1159         <!-- Contains the license agreement in plain text. Several
1160           different locales can be specified -->
1161         <xs:element name="License" type="ovf:Info_Type"
1162           minOccurs="1" maxOccurs="unbounded"/>
1163         <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
1164 maxOccurs="unbounded"/>
1165         <xs:any namespace="##other" processContents="lax" minOccurs="0"
1166 maxOccurs="unbounded"/>
1167       </xs:sequence>
1168       <xs:anyAttribute namespace="##any"/>
1169     </xs:extension>
1170   </xs:complexContent>
1171 </xs:complexType>
1172
1173 <!-- For a VirtualSystemCollection, this section is used to specify the order in which the
1174   contained entities are to be powered on.
1175 -->
1176 <xs:complexType name="StartupSection_Type">
1177   <xs:complexContent>
1178     <xs:extension base="ovf:Section_Type">
1179       <xs:sequence>
1180         <xs:element name="item" minOccurs="0" maxOccurs="unbounded">
1181           <xs:complexType>
1182             <!-- Id of entity in collection -->
1183             <xs:attribute name="id" type="xs:string"/>
1184             <!-- Startup order. Entities are started up starting with lower-numbers
1185               first. Items with same order identifier may be started up
1186               concurrently or in any order. The order is reversed for shutdown.
1187             -->
1188             <xs:attribute name="order" type="xs:int"/>
1189             <!-- Delay in seconds to wait for the power on to complete -->
1190             <xs:attribute name="startDelay" type="xs:int"/>
1191             <!-- Whether to resume power-on sequence, once the guest reports ok. -->
1192             <xs:attribute name="waitingForGuest" type="xs:boolean"/>
1193             <!-- Delay in seconds to wait for the power on to complete -->
1194             <xs:attribute name="stopDelay" type="xs:int"/>
1195             <!-- Stop action to use. Valid values are: 'powerOn' (default), 'none'. -->
1196             <xs:attribute name="startAction" type="xs:string"/>
1197             <!-- Stop action to use. Valid values are: 'powerOff' (default),
1198               'guestShutdown', 'suspend'.
1199             -->
1200             <xs:attribute name="stopAction" type="xs:string"/>
1201             <xs:anyAttribute namespace="##any"/>

```

```

1202         </xs:complexType>
1203     </xs:element>
1204     <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
1205 maxOccurs="unbounded" />
1206     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1207 maxOccurs="unbounded" />
1208 </xs:sequence>
1209 <!-- A comma-separated list of transports that the virtual
1210 machine supports to provide feedback.
1211 -->
1212     <xs:anyAttribute namespace="##any" />
1213 </xs:extension>
1214 </xs:complexContent>
1215 </xs:complexType>
1216
1217 <!-- If this section is present, it indicates that the virtual machine
1218 needs to be initially booted to install and configure the software.
1219 -->
1220 <xs:complexType name="InstallSection_Type">
1221     <xs:complexContent>
1222         <xs:extension base="ovf:Section_Type">
1223             <xs:sequence>
1224                 <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
1225 maxOccurs="unbounded" />
1226                 <xs:any namespace="##other" processContents="lax" minOccurs="0"
1227 maxOccurs="unbounded" />
1228             </xs:sequence>
1229             <!-- A comma-separated list of transports that the virtual
1230 machine supports to provide feedback.
1231 -->
1232             <xs:attribute name="transport" type="xs:string" />
1233             <xs:anyAttribute namespace="##any" />
1234         </xs:extension>
1235     </xs:complexContent>
1236 </xs:complexType>
1237 </xs:schema>
1238
1239

```


1240 **Schema: ovf-virtualhardware.xsd**

1241 The schema below describes the XML representation of the virtual hardware.

```

1242 <?xml version="1.0" encoding="UTF-8"?>
1243 <xs:schema
1244     targetNamespace="http://schemas.dmtf.org/ovf/envelope"
1245     xmlns:ovf="http://schemas.dmtf.org/ovf/envelope"
1246     xmlns:xs="http://www.w3.org/2001/XMLSchema"
1247     xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
1248     xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData">
1249     <xs:import namespace="http://www.w3.org/XML/1998/namespace"
1250         schemaLocation="http://www.w3.org/2001/xml.xsd" />
1251     <xs:include schemaLocation="ovf-section.xsd" />
1252
1253     <xs:import
1254         namespace="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
1255         schemaLocation="cim-vssd.xsd" />
1256     <xs:import
1257         namespace="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
1258         schemaLocation="cim-rasd.xsd" />
1259
1260     <!-- Specifies the virtual hardware for a virtual machine -->
1261     <xs:complexType name="VirtualHardwareSection_Type">
1262         <xs:complexContent>
1263             <xs:extension base="ovf:Section_Type">
1264                 <xs:sequence>
1265                     <xs:element name="System"
1266                         type="vssd:CIM_VirtualSystemSettingData_Type"
1267                         minOccurs="0" />
1268                     <xs:element name="Item"
1269                         type="rasd:CIM_ResourceAllocationSettingData_Type"
1270                         minOccurs="0" maxOccurs="unbounded" />
1271                 </xs:sequence>
1272             </xs:extension>
1273         </xs:complexContent>
1274     </xs:complexType>
1275
1276     <!-- Specifies a section for resource constraints on a VirtualSystemCollection -->
1277     <xs:complexType name="ResourceAllocationSection_Type">
1278         <xs:complexContent>
1279             <xs:extension base="ovf:Section_Type">
1280                 <xs:sequence>
1281                     <xs:element name="Item"
1282                         type="rasd:CIM_ResourceAllocationSettingData_Type"
1283                         minOccurs="0" maxOccurs="unbounded" />
1284                 </xs:sequence>
1285             </xs:extension>
1286         </xs:complexContent>
1287     </xs:complexType>
1288 </xs:schema>

```

1289

1290 **Schema: cim-rasd.xsd**

1291 The schema below describes the XML representation of the CIM ResourceAllocationSettingData .

```

1292 <?xml version='1.0' encoding='utf-8'?>
1293 <xs:schema
1294     targetNamespace="http://schemas.dmtf.org/wbem/wscim/1/cim-
1295     schema/2/CIM_ResourceAllocationSettingData"
1296     xmlns:class="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
1297     xmlns:cim="http://schemas.dmtf.org/wbem/wscim/1/common"
1298     xmlns:xs="http://www.w3.org/2001/XMLSchema">
1299
1300     <xs:import namespace="http://schemas.dmtf.org/wbem/wscim/1/common" schemaLocation="cim-
1301     common.xsd"/>
1302
1303     <xs:element name="Caption" nillable="true" type="cim:cimString"/>
1304
1305     <xs:element name="Description" nillable="true" type="cim:cimString"/>
1306
1307     <xs:element name="InstanceId" nillable="true" type="cim:cimString"/>
1308
1309     <xs:element name="ResourceType" nillable="true">
1310         <xs:complexType>
1311             <xs:simpleContent>
1312                 <xs:restriction base="xs:anyType">
1313                     <xs:simpleType>
1314                         <xs:union>
1315                             <xs:simpleType>
1316                                 <xs:restriction base="xs:unsignedShort">
1317                                     <xs:enumeration value="1"/> <!-- Other -->
1318                                     <xs:enumeration value="2"/> <!-- Computer System -->
1319                                     <xs:enumeration value="3"/> <!-- Processor-->
1320                                     <xs:enumeration value="4"/> <!-- Memory-->
1321                                     <xs:enumeration value="5"/> <!-- IDE Controller -->
1322                                     <xs:enumeration value="6"/> <!-- Parallel SCSI HBA -->
1323                                     <xs:enumeration value="7"/> <!-- FC HBA -->
1324                                     <xs:enumeration value="8"/> <!-- iSCSI HBA -->
1325                                     <xs:enumeration value="9"/> <!-- IB HCA -->
1326                                     <xs:enumeration value="10"/> <!-- Ethernet Adapter -->
1327                                     <xs:enumeration value="11"/> <!-- Other Network Adapter -->
1328                                     <xs:enumeration value="12"/> <!-- I/O Slot -->
1329                                     <xs:enumeration value="13"/> <!-- I/O Device -->
1330                                     <xs:enumeration value="14"/> <!-- Floppy Drive -->
1331                                     <xs:enumeration value="15"/> <!-- CD Drive -->
1332                                     <xs:enumeration value="16"/> <!-- DVD drive -->
1333                                     <xs:enumeration value="17"/> <!-- Disk Drive -->
1334                                     <xs:enumeration value="18"/> <!-- Tape Drive -->
1335                                     <xs:enumeration value="19"/> <!-- Storage Extent -->
1336                                     <xs:enumeration value="20"/> <!-- Other storage device -->
1337                                     <xs:enumeration value="21"/> <!-- Serial port -->
1338                                     <xs:enumeration value="22"/> <!-- Parallel port -->
1339                                     <xs:enumeration value="23"/> <!-- USB Controller -->
1340                                     <xs:enumeration value="24"/> <!-- Graphics controller -->
1341                                     <xs:enumeration value="25"/> <!-- IEEE 1394 Controller -->
1342                                     <xs:enumeration value="26"/> <!-- Partitionable Unit -->
1343                                     <xs:enumeration value="27"/> <!-- Base Partitionable Unit -->
1344                                     <xs:enumeration value="28"/> <!-- Power Supply -->
1345                                     <xs:enumeration value="29"/> <!-- Cooling Device -->
1346                                     <xs:enumeration value="30"/> <!-- PCI Controller -->
1347                                     <xs:enumeration value="31"/> <!-- PS2 Controller -->
1348                                     <xs:enumeration value="32"/> <!-- SIO Controller -->
1349                                     <xs:enumeration value="33"/> <!-- Keyboard -->
1350                                     <xs:enumeration value="34"/> <!-- Pointing Device -->
1351                                 </xs:restriction>
1352                             </xs:simpleType>
1353                         </xs:union>
1354                     </xs:simpleType>
1355                 </xs:restriction>
1356             </xs:simpleContent>
1357         </xs:complexType>
1358     </xs:element>

```

```

1354         <xs:restriction base="xs:unsignedShort">
1355             <xs:minInclusive value="30"/>
1356             <xs:maxInclusive value="32769"/>
1357         </xs:restriction>
1358     </xs:simpleType>
1359 </xs:simpleType>
1360     <xs:restriction base="xs:unsignedShort">
1361         <xs:minInclusive value="32768"/>
1362         <xs:maxInclusive value="65535"/>
1363     </xs:restriction>
1364 </xs:simpleType>
1365 </xs:union>
1366 </xs:simpleType>
1367     <xs:anyAttribute namespace="##any"/>
1368 </xs:restriction>
1369 </xs:simpleContent>
1370 </xs:complexType>
1371 </xs:element>
1372
1373 <xs:element name="OtherResourceType" nillable="true" type="cim:cimString"/>
1374
1375 <xs:element name="ResourceSubType" nillable="true" type="cim:cimString"/>
1376
1377 <xs:element name="PoolID" nillable="true" type="cim:cimString"/>
1378
1379 <xs:element name="ConsumerVisibility" nillable="true">
1380     <xs:complexType>
1381         <xs:simpleContent>
1382             <xs:restriction base="xs:anyType">
1383                 <xs:simpleType>
1384                     <xs:union>
1385                         <xs:simpleType>
1386                             <xs:restriction base="xs:unsignedShort">
1387                                 <xs:enumeration value="0"/>
1388                                 <xs:enumeration value="2"/>
1389                                 <xs:enumeration value="3"/>
1390                                 <xs:enumeration value="4"/>
1391                             </xs:restriction>
1392                         </xs:simpleType>
1393                     <xs:simpleType>
1394                         <xs:restriction base="xs:unsignedShort">
1395                             <xs:minInclusive value="5"/>
1396                             <xs:maxInclusive value="32768"/>
1397                         </xs:restriction>
1398                     </xs:simpleType>
1399                 </xs:simpleType>
1400             <xs:restriction base="xs:unsignedShort">
1401                 <xs:minInclusive value="32767"/>
1402                 <xs:maxInclusive value="65535"/>
1403             </xs:restriction>
1404         </xs:simpleType>
1405     </xs:union>
1406 </xs:simpleType>
1407     <xs:anyAttribute namespace="##any"/>
1408 </xs:restriction>
1409 </xs:simpleContent>
1410 </xs:complexType>
1411 </xs:element>
1412
1413 <xs:element name="HostResource" nillable="true" type="xs:anyType"/>
1414
1415 <xs:element name="AllocationUnits" nillable="true" type="cim:cimString"/>
1416
1417 <xs:element name="VirtualQuantity" nillable="true"
1418     type="cim:cimUnsignedLong"/>
1419
1420 <xs:element name="Reservation" nillable="true" type="cim:cimUnsignedLong"/>
1421
1422 <xs:element name="Limit" nillable="true" type="cim:cimUnsignedLong"/>
1423

```

```

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"/>

```

```
1494     <xs:element ref="class:Parent" minOccurs="0"/>
1495     <xs:element ref="class:Connection" maxOccurs="unbounded" minOccurs="0"/>
1496     <xs:element ref="class:Address" minOccurs="0"/>
1497     <xs:element ref="class:MappingBehavior" minOccurs="0"/>
1498     <xs:element ref="class:AddressOnParent" minOccurs="0"/>
1499     <xs:element ref="class:BusNumber" minOccurs="0"/>
1500     <xs:any namespace="##other" processContents="lax" minOccurs="0"
1501           maxOccurs="unbounded"/>
1502   </xs:sequence>
1503   <xs:anyAttribute namespace="##any"/>
1504 </xs:complexType>
1505
1506   <xs:element name="CIM_ResourceAllocationSettingData"
1507             type="class:CIM_ResourceAllocationSettingData_Type"/>
1508 </xs:schema>
```

1509 **Schema: cim-vssd.xsd**

1510 The schema below describes the XML representation of the CIM VirtualSystemSettingData.

```

1511 <?xml version='1.0' encoding='utf-8'?>
1512 <xs:schema
1513     targetNamespace="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
1514     xmlns:class="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
1515     xmlns:cim="http://schemas.dmtf.org/wbem/wscim/1/common"
1516     xmlns:xs="http://www.w3.org/2001/XMLSchema">
1517     <xs:import namespace="http://schemas.dmtf.org/wbem/wscim/1/common"
1518         schemaLocation="cim-common.xsd" />
1519
1520     <xs:element name="Caption" nillable="true" type="cim:cimString"/>
1521
1522     <xs:element name="Description" nillable="true" type="cim:cimString"/>
1523
1524     <xs:element name="InstanceId" nillable="true" type="cim:cimString"/>
1525
1526     <xs:element name="VirtualSystemIdentifier" nillable="true" type="cim:cimString"/>
1527
1528     <xs:element name="VirtualSystemType" nillable="true" type="cim:cimString"/>
1529
1530     <xs:complexType name="CIM_VirtualSystemSettingData_Type">
1531         <xs:sequence>
1532             <xs:element ref="class:Caption" minOccurs="0" maxOccurs="unbounded"/>
1533             <xs:element ref="class:Description" minOccurs="0" maxOccurs="unbounded"/>
1534             <xs:element ref="class:InstanceId" minOccurs="0"/>
1535             <xs:element ref="class:VirtualSystemIdentifier" minOccurs="0"/>
1536             <xs:element ref="class:VirtualSystemType" minOccurs="0"/>
1537             <xs:any namespace="##other" processContents="lax" minOccurs="0"
1538                 maxOccurs="unbounded"/>
1539         </xs:sequence>
1540         <xs:anyAttribute namespace="##any"/>
1541     </xs:complexType>
1542
1543     <xs:element name="CIM_VirtualSystemSettingData"
1544         type="class:CIM_VirtualSystemSettingData_Type" />
1545 </xs:schema>
1546
1547

```

1548 **Schema: cim-common.xsd**

1549 The schema below describes the common CIM types.

```

1550 <?xml version="1.0" encoding="utf-8"?>
1551 <xs:schema targetNamespace="http://schemas.dmtf.org/wbem/wscim/1/common"
1552   xmlns:cim="http://schemas.dmtf.org/wbem/wscim/1/common"
1553   xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
1554
1555   <!-- The following are runtime attribute definitions -->
1556   <xs:attribute name="Key" type="xs:boolean"/>
1557
1558   <xs:attribute name="Version" type="xs:string"/>
1559
1560   <!-- The following section defines the extended WS-CIM datatypes -->
1561   <xs:complexType name="cimDateTime">
1562     <xs:choice>
1563       <xs:element name="CIM_DateTime" type="xs:string" nillable="true"/>
1564       <xs:element name="Interval" type="xs:duration"/>
1565       <xs:element name="Date" type="xs:date"/>
1566       <xs:element name="Time" type="xs:time"/>
1567       <xs:element name="Datetime" type="xs:dateTime"/>
1568     </xs:choice>
1569     <xs:anyAttribute namespace="##any" processContents="lax"/>
1570   </xs:complexType>
1571
1572   <xs:complexType name="cimUnsignedByte">
1573     <xs:simpleContent>
1574       <xs:extension base="xs:unsignedByte">
1575         <xs:anyAttribute namespace="##any" processContents="lax"/>
1576       </xs:extension>
1577     </xs:simpleContent>
1578   </xs:complexType>
1579
1580
1581   <xs:complexType name="cimByte">
1582     <xs:simpleContent>
1583       <xs:extension base="xs:byte">
1584         <xs:anyAttribute namespace="##any" processContents="lax"/>
1585       </xs:extension>
1586     </xs:simpleContent>
1587   </xs:complexType>
1588
1589   <xs:complexType name="cimUnsignedShort">
1590     <xs:simpleContent>
1591       <xs:extension base="xs:unsignedShort">
1592         <xs:anyAttribute namespace="##any" processContents="lax"/>
1593       </xs:extension>
1594     </xs:simpleContent>
1595   </xs:complexType>
1596
1597   <xs:complexType name="cimShort">
1598     <xs:simpleContent>
1599       <xs:extension base="xs:short">
1600         <xs:anyAttribute namespace="##any" processContents="lax"/>
1601       </xs:extension>
1602     </xs:simpleContent>
1603   </xs:complexType>
1604
1605   <xs:complexType name="cimUnsignedInt">
1606     <xs:simpleContent>
1607       <xs:extension base="xs:unsignedInt">
1608         <xs:anyAttribute namespace="##any" processContents="lax"/>
1609       </xs:extension>
1610     </xs:simpleContent>
1611   </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">

```



```

1682         <xs:anyAttribute namespace="##any" processContents="lax"/>
1683     </xs:extension>
1684 </xs:simpleContent>
1685 </xs:complexType>
1686
1687 <xs:complexType name="cimHexBinary">
1688     <xs:simpleContent>
1689         <xs:extension base="xs:hexBinary">
1690             <xs:anyAttribute namespace="##any" processContents="lax"/>
1691         </xs:extension>
1692     </xs:simpleContent>
1693 </xs:complexType>
1694
1695 <xs:complexType name="cimReference">
1696     <xs:sequence>
1697         <xs:any namespace="##other" maxOccurs="unbounded"/>
1698     </xs:sequence>
1699     <xs:anyAttribute namespace="##any" processContents="lax"/>
1700 </xs:complexType>
1701
1702 <!-- The following datatypes are used exclusively to define metadata fragments -->
1703 <xs:attribute name="qualifier" type="xs:boolean"/>
1704
1705 <xs:complexType name="qualifierString">
1706     <xs:simpleContent>
1707         <xs:extension base="cim:cimString">
1708             <xs:attribute ref="cim:qualifier" use="required"/>
1709         </xs:extension>
1710     </xs:simpleContent>
1711 </xs:complexType>
1712
1713 <xs:complexType name="qualifierBoolean">
1714     <xs:simpleContent>
1715         <xs:extension base="cim:cimBoolean">
1716             <xs:attribute ref="cim:qualifier" use="required"/>
1717         </xs:extension>
1718     </xs:simpleContent>
1719 </xs:complexType>
1720
1721 <xs:complexType name="qualifierUInt32">
1722     <xs:simpleContent>
1723         <xs:extension base="cim:cimUnsignedInt">
1724             <xs:attribute ref="cim:qualifier" use="required"/>
1725         </xs:extension>
1726     </xs:simpleContent>
1727 </xs:complexType>
1728
1729 <xs:complexType name="qualifierSInt64">
1730     <xs:simpleContent>
1731         <xs:extension base="cim:cimLong">
1732             <xs:attribute ref="cim:qualifier" use="required"/>
1733         </xs:extension>
1734     </xs:simpleContent>
1735 </xs:complexType>
1736
1737 <xs:complexType name="qualifierSArray">
1738     <xs:complexContent>
1739         <xs:extension base="cim:qualifierString"/>
1740     </xs:complexContent>
1741 </xs:complexType>
1742
1743 <!-- The following element is to be used only for defining metadata -->
1744 <xs:element name="DefaultValue" type="xs:anySimpleType"/>
1745 </xs:schema>
1746
1747

```

1748 **Schema: ovf-environment.xsd**

1749 The schema below describes the XML representation of the OVF environment.

```

1750 <?xml version="1.0" encoding="UTF-8"?>
1751 <xs:schema
1752     targetNamespace="http://schemas.dmtf.org/ovf/environment"
1753     xmlns:ovfenv="http://schemas.dmtf.org/ovf/environment"
1754     xmlns:xs="http://www.w3.org/2001/XMLSchema">
1755
1756     <!-- Root element of a OVF environment -->
1757     <xs:element name="Environment">
1758         <xs:complexType>
1759             <xs:sequence>
1760                 <!-- Entity independent sections -->
1761                 <xs:element name="Section" type="ovfenv:Section_Type" minOccurs="0"
1762                     maxOccurs="unbounded" />
1763
1764                 <!-- Information particular to an entity -->
1765                 <xs:element name="Entity" type="ovfenv:Entity_Type" minOccurs="0"
1766                     maxOccurs="unbounded" />
1767             </xs:sequence>
1768             <xs:attribute name="version" type="xs:string" use="required" />
1769             <xs:attribute name="id" type="xs:string" use="required" />
1770         </xs:complexType>
1771     </xs:element>
1772
1773     <!-- Abstract type for all sections -->
1774     <xs:complexType name="Section_Type" abstract="true" />
1775
1776     <!-- An entity is a container of sections for a given entity -->
1777     <xs:complexType name="Entity_Type">
1778         <xs:sequence>
1779             <xs:element name="Section" type="ovfenv:Section_Type" minOccurs="0"
1780                 maxOccurs="unbounded" />
1781         </xs:sequence>
1782         <xs:attribute name="id" type="xs:string" use="required" />
1783     </xs:complexType>
1784
1785     <!-- The key/values of assigned properties for an entity -->
1786     <xs:complexType name="Property_Type">
1787         <xs:complexContent>
1788             <xs:extension base="ovfenv:Section_Type">
1789                 <xs:sequence>
1790                     <xs:element name="Property"
1791                         minOccurs="0" maxOccurs="unbounded">
1792                         <xs:complexType>
1793                             <xs:attribute name="key" type="xs:string" />
1794                             <xs:attribute name="value" type="xs:string" />
1795                         </xs:complexType>
1796                     </xs:element>
1797                 </xs:sequence>
1798             </xs:extension>
1799         </xs:complexContent>
1800     </xs:complexType>
1801
1802     <!-- Information about deployment platform -->
1803     <xs:complexType name="Platform_Type">
1804         <xs:complexContent>
1805             <xs:extension base="ovfenv:Section_Type">
1806                 <xs:sequence>
1807                     <xs:element name="kind" type="xs:string" />
1808                     <xs:element name="version" type="xs:string" />
1809                     <xs:element name="vendor" type="xs:string" />
1810                     <xs:element name="local" type="xs:string" />
1811                 </xs:sequence>

```

```
1812         </xs:extension>
1813     </xs:complexContent>
1814 </xs:complexType>
1815 </xs:schema>
```

1816

ANNEX C: CIM MOFs

1817

MOF: CIM_VirtualSystemSettingData

```

1818 // =====
1819 // CIM_VirtualSystemSettingData
1820 // =====
1821 [ Experimental,
1822   Version("2.14.0"),
1823   Description(
1824     "CIM_VirtualSystemSettingData defines the virtual aspects of a "
1825     "virtual system through a set of virtualization specific "
1826     "properties. CIM_VirtualSystemSettingData is also used as the "
1827     "top level class of virtual system configurations. Virtual "
1828     "system configurations model configuration information about "
1829     "virtual systems and their components. A virtual system "
1830     "configuration consists of one top-level instance of class "
1831     "CIM_VirtualSystemSettingData that aggregates a number of "
1832     "instances of class CIM_ResourceAllocationSettingData, using "
1833     "association CIM_ConcreteComponent.\n"
1834     "Virtual system configurations may for example be used to "
1835     "reflect configurations of\n"
1836     "- virtual systems that are defined at a virtualization "
1837     "platform,\n"
1838     "- virtual systems that are currently active,\n"
1839     "- input requests to create new virtual systems,\n"
1840     "- input requests to modify existing virtual systems, or\n"
1841     "- snapshots of virtual systems.")
1842 class CIM_VirtualSystemSettingData : CIM_SettingData {
1843
1844   [ Description(
1845     "VirtualSystemIdentifier shall reflect a unique name for "
1846     "the system as it is used within the virtualization platform. "
1847     "Note that the VirtualSystemIdentifier is not the hostname "
1848     "assigned to the operating system instance running within the "
1849     "virtual system, nor is it an IP address or MAC address "
1850     "assigned to any of its network ports.\n"
1851     "On create requests VirtualSystemIdentifier may contain "
1852     "implementation specific rules (like simple patterns or "
1853     "regular expression) that may be interpreted by the "
1854     "implementation when assigning a VirtualSystemIdentifier."]
1855   string VirtualSystemIdentifier;
1856
1857   [ Description(
1858     "VirtualSystemType shall reflect a particular type of virtual "
1859     "system. Implementations are commonly capable to support "
1860     "various implementation defined virtual system types.\n"
1861     "As stated in the class description, instances of this class "
1862     "may be used for various purposes. A management application "
1863     "intending to use an instance of this class as input parameter "
1864     "to an operation that creates or modifies a virtual system "
1865     "should first determine the set of valid virtual system types "
1866     "that are supported by the virtualization platform hosting the "
1867     "virtual system by inspecting values of array property "
1868     "VirtualSystemTypesSupported of the instance of class "
1869     "CIM_VirtualSystemManagementCapabilities that describes the "
1870     "capabilities of the virtualization platform.")
1871   ModelCorrespondence(
1872     "CIM_VirtualSystemManagementCapabilities.VirtualSystemTypesSupported"])
1873   string VirtualSystemType;
1874 };

```

1875 **MOF: CIM_ResourceAllocationSettingData**

1876 The CIM_ResourceAllocationSettingData is currently a Change Request (CR) to CIM. In this specification, we propose a
 1877 few changes to the CR in order to capture all attributes of virtual hardware. These additions are shown in **bold face on**
 1878 **grey background**.

```

1879
1880 // Copyright (c) 2007 DMTF. All rights reserved.
1881 // =====
1882 // CIM_ResourceAllocationSettingData
1883 // =====
1884 [Experimental, Version ("2.16.0"), Description (
1885     "The ResourceAllocationSettingData class represents settings "
1886     "specifically related to an allocated resource that are outside "
1887     "the scope of the CIM class typically used to represent the "
1888     "resource itself. These settings include information specific "
1889     "to the allocation that may not be visible to the consumer "
1890     "of the resource itself. For example, a virtual processor may look like "
1891     "a 2 ghz processor to the consumer (virtual computer system), however "
1892     "the virtualization system may use time-slicing to schedule the "
1893     "the virtual processor to only allow it to use 1 ghz. ")] class CIM_ResourceAllocationSettingData
1894 : CIM_SettingData {
1895
1896     [Description ("The type of resource this allocation setting represents."),
1897     ValueMap { "1", "2", "3", "4", "5", "6", "7", "8", "9",
1898         "10", "11", "12", "13", "14", "15", "16", "17", "18", "19",
1899         "20", "21", "22", "23", "24", "25", "26", "27", "28", "29",
1900         "30", "31", "32", "33", "34",
1901         "..", "0x8000..0xFFFF" },
1902     Values { "Other", "Computer System", "Processor", "Memory",
1903         "IDE Controller", "Parallel SCSI HBA", "FC HBA",
1904         "iSCSI HBA", "IB HCA", "Ethernet Adapter",
1905         "Other Network Adapter", "I/O Slot", "I/O Device",
1906         "Floppy Drive", "CD Drive", "DVD drive", "Disk Drive",
1907         "Tape Drive", "Storage Extent", "Other storage device", "Serial port",
1908         "Parallel port", "USB Controller", "Graphics controller",
1909         "IEEE 1394 Controller", "Partitionable Unit",
1910         "Base Partitionable Unit", "Power Supply", "Cooling Device",
1911         "PS2 Controller", "SIO Controller", "Keyboard", "Pointing Device",
1912         "PCI Controller"
1913         "DMTF reserved", "Vendor Reserved"},
1914     ModelCorrespondence {
1915         "CIM_ResourceAllocationSettingData.OtherResourceType",
1916         "CIM_ResourceAllocationSettingData.ResourceSubType"}}
1917     uint16 ResourceType;
1918
1919     [Description ("A string that describes the resource type when a "
1920     "well defined value is not available and Resource type has "
1921     "the value \"Other\"."),
1922     ModelCorrespondence {
1923         "CIM_ResourceAllocationSettingData.ResourceType" }]
1924     string OtherResourceType;
1925
1926     [Description ("A string describing an implementation specific "
1927     "sub-type for this resource. For example, this may "
1928     "be used to distinguish different models of the same "
1929     "resource type."),
1930     ModelCorrespondence {
1931         "CIM_ResourceAllocationSettingData.ResourceType" }]
1932     string ResourceSubType;
1933
1934     [Description ("This property specifies which ResourcePool the "
1935     "resource is currently allocated from, or which ResourcePool "
1936     "the resource will be allocated from when the allocation "
1937     "occurs."),
1938     ModelCorrespondence { "CIM_ResourcePool.PoolId" }]
1939     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

```

```

2010     [Description ("This property specifies a relative priority for this "
2011     "allocation in relation to other allocations from the same "
2012     "ResourcePool. This property has no unit of measure, and is "
2013     "only relevant when compared to other allocations vying for "
2014     "the same host resources.")]
2015     uint32 Weight;
2016
2017     [Description ("This property specifies if the resource will be "
2018     "automatically allocated. For example when set to true, "
2019     "when the consuming virtual computer system is powered on, "
2020     "this resource would be allocated. A value of false "
2021     "indicates the resource must be explicitly allocated. For "
2022     "example, the setting may represent removable media "
2023     "(cdrom, floppy, etc.) where at power on time, the media is"
2024     "not present. An explicit operation is required to allocate "
2025     "the resource.")]
2026     boolean AutomaticAllocation;
2027
2028     [Description ("This property specifies if the resource will be "
2029     "automatically de-allocated. For example, when set to true, "
2030     "when the consuming virtual computer system is powered off, "
2031     "this resource would be de-allocated. When set to false, the "
2032     "resource will remain allocated and must be explicitly "
2033     "de-allocated.")]
2034     boolean AutomaticDeallocation;
2035
2036     [Description ("The Parent of the resource."
2037     "For example, a controller for the current allocation")]
2038     string Parent;
2039
2040     [Description ("The thing to which this resource is connected. "
2041     "For example, a named network or switch port.")]
2042     string Connection[];
2043
2044     [Description ("The address of the resource."
2045     "For example, the MAC address of a Ethernet port.")]
2046     string Address;
2047
2048     [Description ("Specifies how this resource maps to underlying resources "
2049     "If the HostResource array contains any entries, this property "
2050     "reflects how the resource maps to those specific resources. "),
2051     ValueMap { "0", "1", "2", "3", "4", "..", "32767..65535" },
2052     Values { "Unknown", "Not Supported", "Dedicated",
2053     "Soft Affinity", "Hard Affinity", "DMTF Reserved", "Vendor Reserved" }]
2054     uint16 MappingBehavior;
2055
2056     [Description("Describes the address of this resource in the context "
2057     "of the Parent. For example, if the parent is a PCI Controller "
2058     "this property would specify the PCI slot of the child device.")]
2059     string AddressOnParent;
2060
2061     [Description("Specifies the bus on the controller that this describes. "
2062     " For example, an IDE controller implements multiple buses.")]
2063     uint16 BusNumber;
2064
2065 };
2066

```

2067 **ANNEX C: Authors and Acknowledgements**

2068 This specification has been developed as a result of joint work with many individuals and teams, including:

2069 [Xen Source](#)

2070 [VMware, Inc.](#)

2071 [IBM](#)

2072 [HP](#)

2073 [Dell](#)

2074 [Microsoft](#)

2075 Bibliography
2076 This section contains a list of the external references and dependencies for this specification. Dependencies are
2077 normative. References are non-normative.

2078 **WS-Management**

2079 http://www.dmtf.org/standards/published_documents/DSP0226.pdf

2080 **WS-CIM Mapping Specification**

2081 http://www.dmtf.org/standards/published_documents/DSP0230.pdf

2082 **CIM Schema (MOF files)**

2083 http://www.dmtf.org/standards/cim/cim_schema_v214

2084 **System Virtualization Profile (SVP)**

2085 DMTF DSP1042

2086 **Virtual System Profile (VSP)**

2087 DMTF DSP1057

2088 **Resource Allocation Profile (RAP)**

2089 DMTF DSP1041

2090 http://www.dmtf.org/standards/published_documents/DSP1041.pdf

2091 **Allocation Capabilities Profile (ACP)**

2092 DMTF DSP1043

2093 **HTTP 1.1**

2094 R. Fielding et al, "IETF RFC 2616: Hypertext Transfer Protocol – HTTP/1.1," June 1999.

2095 <http://www.ietf.org/rfc/rfc2616.txt>

2096 **HTTPS**

2097 E. Rescorla, "RFC 2818: HTTP over TLS," May 2000.

2098 <http://www.ietf.org/rfc/rfc2818.txt>

2099 **RFC 2119**

2100 S. Bradner, "RFC 2119: Key words for use in RFCs to Indicate Requirement Levels," March 1997.

2101 <http://www.ietf.org/rfc/rfc2119.txt>

2102 **XML Schema, Part 1**

2103 H. Thompson et al, "XML Schema Part 1: Structures," May 2001.

2104 <http://www.w3.org/TR/xmlschema-1/>

2105 **XML Schema, Part 2**

2106 P. Biron et al, "XML Schema Part 2: Datatypes," May 2001.

2107 <http://www.w3.org/TR/xmlschema-2/>

2108 **RFC 3986: Uniform Resource Identifiers (URI): Generic Syntax**

2109 <http://www.ietf.org/rfc/rfc3986.txt>

2110

2111

2112

2113

2114

2115

2116

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
© 1998-2007 VMware, Inc. All rights reserved. Protected by one or more U.S. Patents Nos. 6,397,242; 6,496,847; 6,704,925; 6,711,672; 6,725,289; 6,735,601; 6,785,886; 6,789,156; 6,795,966; 6,880,022; 6,944,699; 6,961,806; 6,961,941; 7,069,413; 7,082,598; 7,089,377; 7,111,086; 7,111,145; 7,117,481; 7,149,843 and 7,155,558; patents pending. VMware, the VMware "boxes" logo and design, Virtual SMP and VMotion are registered trademarks or trademarks of VMware, Inc. in the United States and/or other jurisdictions. All other marks and names mentioned herein may be trademarks of their respective companies

