



Document Identifier: DSP2017

Date: 2014-04-24

Version: 2.0.0

Open Virtualization Format White Paper

Document Type: DMTF Informational

Document Status: DMTF Informational

Document Language: en-US

Copyright notice

Copyright © 2007, 2014 Distributed Management Task Force, Inc. (DMTF). All rights reserved.

DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability. Members and non-members may reproduce DMTF specifications and documents, provided that correct attribution is given. As DMTF specifications may be revised from time to time, the particular version and release date should always be noted.

Implementation of certain elements of this standard or proposed standard may be subject to third party patent rights, including provisional patent rights (herein "patent rights"). DMTF makes no representations to users of the standard as to the existence of such rights, and is not responsible to recognize, disclose, or identify any or all such third party patent right, owners or claimants, nor for any incomplete or inaccurate identification or disclosure of such rights, owners or claimants. DMTF shall have no liability to any party, in any manner or circumstance, under any legal theory whatsoever, for failure to recognize, disclose, or identify any such third party patent rights, or for such party's reliance on the standard or incorporation thereof in its product, protocols or testing procedures. DMTF shall have no liability to any party implementing such standard, whether such implementation is foreseeable or not, nor to any patent owner or claimant, and shall have no liability or responsibility for costs or losses incurred if a standard is withdrawn or modified after publication, and shall be indemnified and held harmless by any party implementing the standard from any and all claims of infringement by a patent owner for such implementations.

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

Abstract

The Open Virtualization Format (OVF) White Paper describes the application of DSP0243, DSP8023, and DSP8027, the specifications that are part of the Open Virtualization Format (OVF) standard. The intended audience is anyone who wants to understand the OVF package and its application to specific use cases. Some familiarity with virtualization and the general concepts of the CIM model is assumed.

1				Contents	
2	1	Introduction.....			7
3	1.1	Overview.....			7
4	1.2	Design considerations.....			8
5	2	OVF key concepts.....			9
6	2.1	Virtual appliances.....			9
7	2.2	Life cycle.....			10
8	3	XML backgrounder.....			12
9	3.1	Use of XML Schema.....			12
10	3.2	General XML concepts used in OVF.....			13
11	3.2.1	Element.....			13
12	3.2.2	Attribute.....			13
13	3.2.3	Substitution group.....			13
14	4	OVF package.....			14
15	4.1	Structure of an OVF descriptor.....			14
16	4.2	Global attributes defined in OVF Schema.....			14
17	4.3	Extensibility in OVF.....			15
18	4.4	OVF top level elements.....			16
19	4.4.1	VirtualSystem element.....			18
20	4.4.2	VirtualSystemCollection element.....			18
21	4.4.3	References element.....			18
22	4.4.4	DiskSection element.....			18
23	4.4.5	NetworkSection element.....			19
24	4.4.6	DeploymentOptionsSection element.....			22
25	4.4.7	SharedDiskSection element.....			25
26	4.4.8	PlacementGroupSection element.....			27
27	4.5	OVF section elements used in Virtual System and Virtual System Collection.....			27
28	4.5.1	AnnotationSection element.....			27
29	4.5.2	ProductSection element.....			27
30	4.5.3	EulaSection element.....			29
31	4.5.4	VirtualHardwareSection element.....			29
32	4.5.5	PlacementSection element.....			31
33	4.5.6	EncryptionSection element.....			34
34	4.6	OVF section elements used in virtual system collection.....			37
35	4.6.1	ResourceAllocationSection element.....			37
36	4.6.2	StartupSection element.....			38
37	4.6.3	ScaleOutSection element.....			39
38	4.7	OVF section elements used in virtual system.....			41
39	4.7.1	OperatingSystemSection element.....			41
40	4.7.2	InstallSection element.....			42
41	4.7.3	EnvironmentFilesSection element.....			42
42	4.7.4	BootDeviceSection element.....			43
43	5	Authoring an OVF package.....			44
44	5.1	Creation.....			44
45	5.2	Internationalization.....			45
46	5.3	Extensibility.....			46
47	5.3.1	Substitution group.....			47
48	5.3.2	Elements.....			48
49	5.3.3	Attributes.....			49
50	5.4	Conformance.....			49
51	5.5	Virtual hardware description.....			49
52	5.6	Example descriptors.....			51
53	6	Deploying an OVF package.....			51
54	6.1	Deployment.....			51

55 6.2 OVF environment descriptor 52

56 6.3 Resource configuration options during deployment 53

57 6.4 Product customization during deployment using Property elements..... 54

58 7 Portability..... 56

59 ANNEX A (informative) Single virtual system example 58

60 ANNEX B (informative) Multitiered pet store example 61

61 B.1 Architecture and packaging 61

62 B.2 Properties..... 61

63 B.3 Disk layout 63

64 B.4 Pet Store OVF descriptor 63

65 B.5 Complete OVF environment 70

66 ANNEX C (informative) Single virtual system LAMP stack example 72

67 C.1 Deployment-time customization..... 72

68 C.2 Simple LAMP OVF descriptor 75

69 ANNEX D (informative) Multiple virtual system LAMP stack example 79

70 D.1 Two-tier LAMP OVF descriptor 79

71 ANNEX E (informative) Extensibility example..... 85

72 E.1 Custom schema 85

73 E.2 Descriptor with custom extensions 86

74 ANNEX F (informative) Change log 87

75

Figures

77 Figure 1 – OVF package life cycle 10

78 Figure 2 – OVF author function..... 11

79 Figure 3 – OVF deployment function 12

80 Figure 4 – OVF package structure 14

81 Figure 5 – Network connections 20

82 Figure 6 – LAN-SAN network connections 21

83 Figure 7 – Affinity placement 32

84 Figure 8 – Availability placement 33

85 Figure 9 – Affinity and availability placement..... 34

86 Figure B-1 – Pet Store OVF package 61

87 Figure B-2 – Pet Store virtual disk layout..... 63

88

Tables

90 Table B-1 – Web tier configuration 62

91 Table B-2 – Database tier configuration 62

92 Table C-1 – LAMP configuration 72

93

94

95

Foreword

96 The Open Virtualization Format White Paper (DSP2017) was prepared by the OVF Work Group of the
97 DMTF.

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

100	Lawrence Lamers	VMware Inc. (Chair)
101	Marvin Waschke	DMTF Fellow (co-Editor)
102	Peter Wörndle	Ericsson AB (co-Editor)
103	Eric Wells	Hitachi, Ltd. (co-Editor)
104		
105	Hemal Shah	Broadcom Corporation
106	Shishir Pardikar	Citrix Systems Inc.
107	Richard Landau	DMTF Fellow
108	Robert Freund	Hitachi, Ltd.
109	Jeff Wheeler	Huawei
110	Monica Martin	Microsoft Corporation
111	Cheng Wei	Microsoft Corporation
112	Srinivas Maturi	Oracle
113	Steffen Grarup	VMware Inc.
114	Rene Schmidt	VMware Inc.
115	Ghazanfar Ali	ZTE Corporation

116

117

Open Virtualization Format White Paper

118 1 Introduction

119 1.1 Overview

120 The Open Virtualization Format specification (OVF) provides the industry with a standard packaging
121 format for software solutions based on virtual systems, solving critical business needs for software
122 vendors and cloud computing service providers.

123 An OVF package can be used by an independent software vendor (ISV) to publish a software solution; by
124 a data center operator to transport a software solution from one data center to another; by a customer to
125 archive a software solution; or any other use case that can be met by having a standardized package for
126 a software solution.

127 The following use cases are the main basis for OVF work:

- 128 1) the ability for ISVs to package a software solution that is capable of being used on more than
129 one hypervisor
- 130 2) the ability to package a virtual system or collection of virtual systems so they can be moved
131 from one data center to another

132 Other use cases and derivative use cases (i.e., subsets) are also applicable.

133 OVF version 1 has been widely adopted by the industry and is now an international standard.

134 OVF version 2 adds enhanced packaging capabilities, making it applicable to the broader range of use
135 cases that are emerging as industry enters the cloud computing era.

136 OVF 2 adds the following features:

- 137 • Support for Network Ports
- 138 • Scaling at deployment time
- 139 • Support for basic placement policies
- 140 • Encryption of OVF packages
- 141 • Disk sharing at runtime
- 142 • Advanced Device Boot Order
- 143 • Advanced Data Transfer to Guest Software
- 144 • Support for Improved Internationalization - I18N
- 145 • Support of HASH Improved
- 146 • Updated CIM schema

147 OVF has adopted the Common Information Model (CIM), where appropriate, to allow management
148 software to clearly understand and easily map resource properties by using an open standard. The
149 `CIM_ResourceAllocationSettingData` class and its subclasses for specific device types is used to
150 specify the resources needed for the virtual system to operate.

151 OVF 2 supports network configuration and the IEEE Edge Virtual Bridging Discovery and Configuration
152 protocols that use *Network Port Profile* (DSP8049). The `CIM_EthernetPortAllocationSettingData` class
153 provides the essential properties.

154 This document aims to give details of the motivation, goals, design and expected usage of OVF, and
155 should be read in accompaniment with the OVF specification of the same major revision.

156 1.2 Design considerations

157 The rapid adoption of virtual infrastructure has highlighted the need for a standard, portable metadata
158 format for the distribution of virtual systems onto and between virtualization platforms. The ability to
159 package a software application together with the operating system on which it is certified, into a format
160 that can be easily transferred from an ISV, through test and development and into production as a pre-
161 configured, pre-packaged unit with no external dependencies, is extremely attractive. Such pre-deployed,
162 ready to run applications packaged with the configuration of the virtual systems required to run them are
163 called virtual appliances. In order to make this concept practical on a broad scale, it is important that the
164 industry adopts a vendor-neutral standard for packaging such virtual appliances and the metadata
165 required to automatically and securely install, configure, and run them on any virtualization platform.

166 From the user's point of view, OVF is a packaging format for virtual appliances. Once installed, an OVF
167 package adds to the user's infrastructure a self-contained, self-consistent, software application that
168 provides a particular service or services. For example, an OVF package might contain a fully functional
169 and tested web-server, database, and OS combination, such as a LAMP stack (Linux + Apache + MySQL
170 + PHP), or it may contain a virus checker, including its update software, spyware detector, etc.

171 Whereas many virtual appliances contain only a single virtual system, modern enterprise applications are
172 modeled as service oriented architectures (SOA) with multiple tiers, each containing one or more virtual
173 systems. A single virtual system model is thus not sufficient to distribute a multi-tier service. In addition,
174 complex applications require install-time customization of networks and other customer specific
175 properties. Furthermore, a virtual appliance is packaged in a run-time format with disk images and
176 configuration data suitable for a particular hypervisor. Run-time formats are optimized for execution and
177 not for distribution. For efficient software distribution, a number of additional features become critical,
178 including portability, platform independence, verification, signing, versioning, and licensing terms.

179 The OVF specification describes a hypervisor-neutral, efficient, extensible, and open format for the
180 packaging and distribution of virtual appliances composed of one or more virtual systems. It aims to
181 facilitate the automated and secure management not only of individual virtual systems, but also of the
182 virtual appliance as a functional unit.

183 To be successful, OVF has been developed and endorsed by ISVs, virtual appliance vendors, operating
184 system vendors, as well as virtual platform vendors. The OVF specification promotes customer
185 confidence through the collaborative development of common standards for portability and interchange of
186 virtual systems between different vendors' virtualization platforms.

187 OVF is intended to be immediately useful, to solve an immediate business need, and to facilitate the rapid
188 adoption of a common, backwards compatible, yet rich format for packaging virtual appliances.

189 The OVF specification is complimentary to existing IT management standards and frameworks and
190 promotes best-of-breed competition through openness and extensibility. The explicit copyright notice
191 attached to this document is intended to avoid arbitrary, independent, piecewise extensions to the format
192 while permitting free distribution and implementation of the specification.

193

194

195 2 OVF key concepts

196 2.1 Virtual appliances

197 A virtual appliance is a pre-configured software stack comprising one or more virtual systems. Each
198 virtual system is an independently installable run-time entity consisting of an operating system,
199 applications and application-specific data, as well as metadata describing the virtual hardware that is
200 required by the virtual system. Many infrastructure applications and even end-user applications that are
201 accessible over a network, such as a DNS server, a bug tracking database, or a complete CRM solution
202 composed of web, application and database tiers, can be delivered as virtual appliances. Delivering
203 complex software systems and services as a pre-configured software stack can dramatically increase
204 robustness and simplify installation.

205 Virtual appliances are changing the software distribution paradigm because they allow optimization of the
206 software stack for the specific application and to deliver a turnkey service to the end user. For solution
207 providers, building a virtual appliance is simpler and more cost effective than building a hardware
208 appliance. The application is pre-packaged with the operating system that it uses, reducing compatibility
209 testing and certification, allowing the software to be pre-installed in the environment in which it runs – by
210 the ISV that develops the solution. For end users, virtual appliances offer an opportunity to dramatically
211 simplify the software management lifecycle through the adoption of standardized, automated, and
212 efficient processes that replace the individual OS and application specific management tasks used
213 previously.

214 Virtual appliances need not be developed and delivered by third-party ISVs – the concept is equally
215 useful and often used within an enterprise in which a virtual system template for a particular service is
216 assembled, tested, and certified by an IT organization and then packaged for repeated, “cookie cutter”
217 deployment throughout the enterprise.

218 Commonly, a software service is implemented as a multi-tier application running in multiple virtual
219 systems and communicating across the network by using a SOA model. Services are often composed of
220 other services, which themselves might also be multi-tier applications, or composed of other services.
221 Indeed the SOA model naturally fits into a virtual appliance-based infrastructure, because virtual
222 appliances are typified by the use of network facing, XML-based management and service interfaces that
223 allow composition of appliances to deliver a complete application.

224 For example, consider a typical web application that consists of three tiers: a web tier that implements the
225 presentation logic; an application server tier that implements the business logic; and a back-end database
226 tier. A straightforward implementation divides this configuration into three virtual systems, one for each
227 tier. In this way, the application can scale from a fraction of a single physical host to three physical hosts.
228 Another approach is to treat each tier as a service in itself. Hence, each tier can scale to a multi-virtual
229 system service that provides a clustered solution. Again, taking the web application as an example, a
230 common scenario is to have many web servers, fewer applications servers, and one or two database
231 servers. Implemented as virtual systems, each tier can scale across as many or as few physical machines
232 as required, and each tier can support multiple instances of virtual systems that service requests.

233 One OVF may contain a single or many virtual systems. It is left to developers to decide which
234 arrangement best suits their application. OVFs must be installed before they can be run; a particular
235 virtualization platform may run the virtual system from the OVF, but this configuration is not required. If
236 this configuration is chosen, the OVF itself can no longer be viewed as a “golden image” version of the
237 appliance because run-time state for the virtual system(s) pervades the OVF. Moreover the digital
238 signature that allows the platform to check the integrity of the OVF is invalid.

239 As a transport mechanism, OVF differs from VMware's VMDK Virtual Disk Format and Microsoft's VHD
240 Virtual Hard Disk format or the open source QCOW format. These are run-time virtual system image
241 formats, operating at the scope of a single virtual disk. Though they are frequently used as transport
242 formats today, they are not designed to solve portability problems. They do not help with a virtual system

243 that has multiple disks, or multiple virtual systems, or they need customization of the virtual system during
 244 installation. These formats provide no help if the virtual system is intended to run on multiple virtualization
 245 platforms (even if the virtualization platforms claim support of the particular virtual hard disk format used).

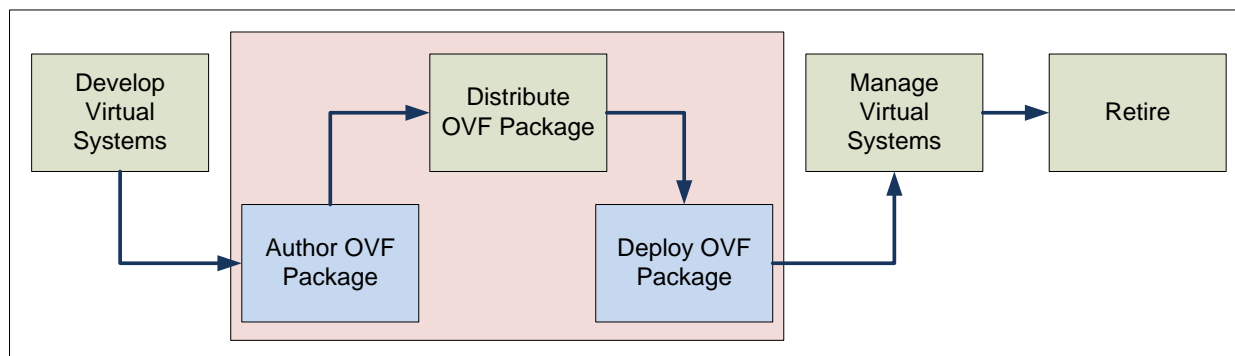
246 Within the OVF remit is the concept of the certification and integrity of a packaged virtual appliance. This
 247 concept allows the platform to determine the provenance of the appliance and permits the end-user to
 248 make the appropriate trust decisions. The OVF specification has been constructed so that the appliance
 249 is responsible for its own configuration and modification. In particular, this means that the virtualization
 250 platform does not need to be able to read from the appliance's file systems. This decoupling of platform
 251 from the appliance means that OVF packages may be implemented by using any operating system, and
 252 installed on any virtualization platform that supports the OVF format. A specific mechanism is provided for
 253 appliances to detect and react to the platform on which they are installed. This mechanism allows
 254 platforms to extend this specification in unique ways without breaking compatibility of appliances across
 255 the industry.

256 The OVF format has several specific features that are designed for complex, multitier services and their
 257 associated distribution, installation, configuration, and execution:

- 258 • Provides direct support for the configuration of multi-tier applications and the composition of
 259 virtual systems to deliver composed services.
- 260 • Permits the specification of both virtual system and application-level configuration.
- 261 • Has robust mechanisms for validation of the contents of the OVF, and full support for
 262 unattended installation to ease the burden of deployment for users, and thereby enhance the
 263 user's experience.
- 264 • Uses commercially accepted procedures for integrity checking of the OVF contents, through the
 265 use of signatures and trusted third parties. This serves to reassure the consumer that an
 266 appliance has not been modified since it signed by the creator of the appliance. This assurance
 267 is seen as critical to the success of the virtual appliance market, and to the viability of
 268 independent creation and online download of appliances.
- 269 • Allows commercial interests of the appliance vendor and user to be respected, by providing a
 270 basic method for presentation and acknowledgement of licensing terms associated with the
 271 appliance.

272 2.2 Life cycle

273 The life cycle for a virtual system is illustrated in Figure 1:



274

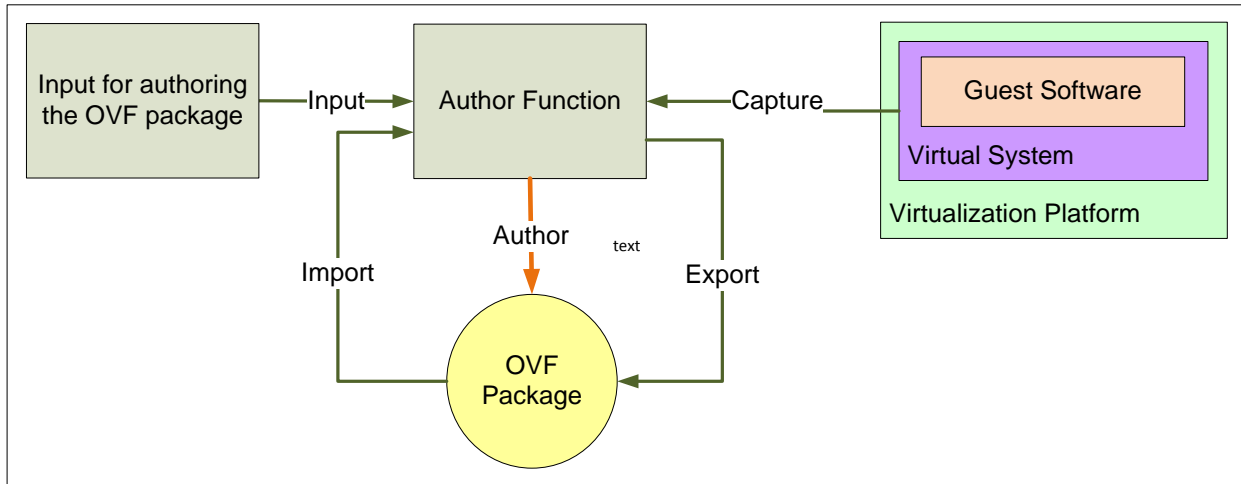
275

Figure 1 – OVF package life cycle

276 An OVF package is built from components that have been developed or acquired by the OVF author.
 277 These are packaged into a set of files that comprise a virtual appliance, consisting of one or more virtual
 278 machines and virtual machine collections and the relevant configuration and deployment metadata. For
 279 example, a clustered database component might be acquired from a third-party ISV. The installed service

280 is then managed and eventually retired. Distribution, management, and retirement are outside the scope
 281 of OVF and are specific to the virtualization product used and the virtual appliance installed from the OVF.
 282 Management includes ongoing maintenance, configuration, and upgrade of the appliance. These
 283 activities depend on the installed service and environment, not the OVF package. The OVF specification
 284 focuses specifically on the authoring and deployment phases.

285 The OVF author function is illustrated in Figure 2.



286

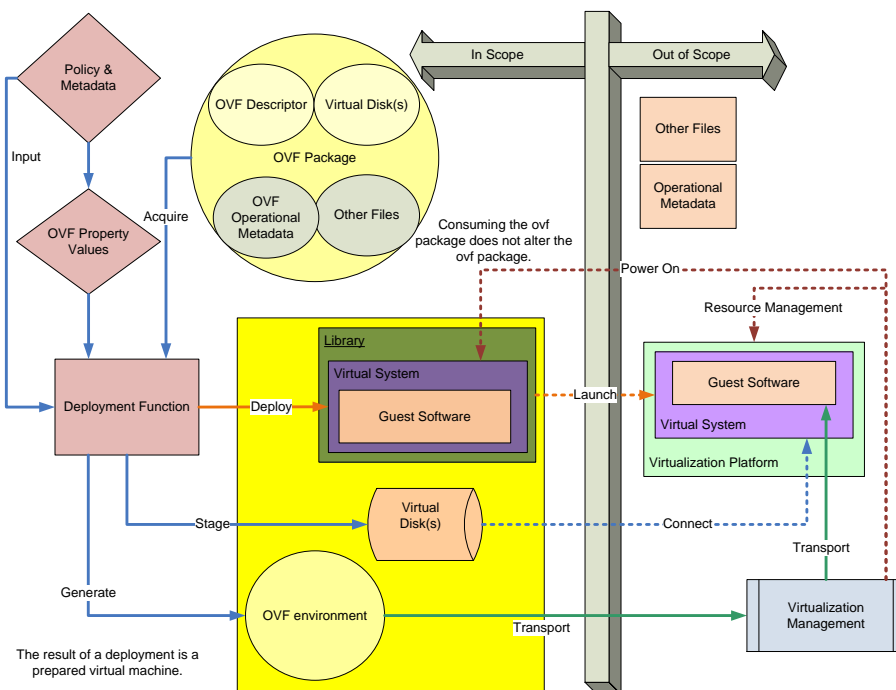
287

Figure 2 – OVF author function

288 An OVF package is authored in one of two ways. The straightforward method is to use a text editor or an
 289 XML authoring tool to create an OVF descriptor, assemble the required disk images and other files, and
 290 then create a tar file or file system that contains the OVF package.

291 An alternative method is to export an OVF package from a virtualization platform. The OVF descriptor
 292 may then be edited to include additional information. This method may be chosen for various reasons,
 293 including improving portability between virtualization platforms or providing options for configuration.

294 The OVF deployment function is illustrated in Figure 3 This diagram is also instructive as to the scope of
 295 work that is covered by the OVF work group.



296

297

Figure 3 – OVF deployment function

298 The OVF operational metadata is information that may be needed for the proper operation of the virtual
 299 system or collection of virtual systems. The OVF operational metadata is a subset of the operational
 300 metadata that may be available when the virtual system is powered on.

301 As shown in the diagram, the OVF deployment function transports the OVF environment to the
 302 virtualization platform. The OVF specification is flexible on the exact nature of the transport, but it can be
 303 thought of as placing media, like a CD ROM, into a virtual reader on the virtual machine and the media
 304 being read by the guest operating system each time the system starts up. The metadata in the OVF
 305 environment is used for configuration after operating system startup and for meeting other requirements
 306 of guest software and virtualization platform for the proper operation of the virtual appliance.

307 **3 XML backgrounder**

308 **3.1 Use of XML Schema**

309 The OVF standard makes use of XML and XML Schema Definition Language (XSD or XSDL). XML is a
 310 markup language that defines a grammar for expressing XML elements and attributes, but says little
 311 about the structure of the elements and attributes in documents or the permissible data types for element
 312 and attribute values. Document Type Definitions (DTDs) have been part of XML from its inception and
 313 were intended to add structure and data types to XML, but they were found to be inadequate in many
 314 situations. In response, the W3C developed XSD, written itself in XML, which is a rich language for
 315 specifying XML document structure and data types. XSD and DTD can be used together, but XSD has
 316 become more common. XSD files are customarily given an “.xsd” extension.

317 An XSD file is metadata that describes the structure and data types of elements and data structures that
 318 are allowed in an XML document. XSD also supports inheritance and other object-oriented constructs.

319 These constructs simplify the creation of elaborate and complex XSDs, making them easier to write, more
320 compact, and have fewer errors. Checking an XML document against the XML grammar only tests the
321 document's syntax, but not the correctness of the structure and data in the document. Validating an XML
322 document against an XSD checks the structure and data. When an XML document is consumed by
323 another process, documents that are valid against an XSD are more likely to be processed without errors.
324 In addition, some tools can generate XML document processors from XSDs. These tools can greatly
325 accelerate development.

326 OVF uses XSD to specify the structure and data types of OVF descriptors, the XML documents in an OVF
327 package that describe arbitrarily complex patterns for the instantiation of virtual systems. OVF descriptors
328 have the file extension ".ovf". The official XSD for OVF 2.0.0 is found at
329 <http://schemas.dmtf.org/ovf/envelope/2/dsp8023.xsd>. Users of OVF should use an XSD validation tool to
330 check their OVF descriptors against the OVF XSD schema. A successfully validated descriptor is a
331 necessary, but not sufficient, condition for OVF standard conformance. Some constraints in the standard
332 specification go beyond the dsp8023.xsd and these must be checked manually for full conformance.

333 **3.2 General XML concepts used in OVF**

334 **3.2.1 Element**

335 An XML element is a data container in the OVF descriptor. Each element that is not empty begins with a
336 start tag, followed by the elements contents, and the element is then closed with an end tag. The start tag
337 consists of the element name, optionally followed by element attributes and surrounded by angled
338 brackets, i.e., <elementname> or <elementname attributes>. The end tag consists of only the element
339 name preceded by a / and enclosed in angle brackets, i.e., </elementname>. An element can contain
340 other elements, text, attributes, or a combination of all of these.

341 XML elements follow these naming rules:

- 342 • Names can contain letters, numbers, and other characters.
- 343 • Names cannot start with a number or punctuation character.
- 344 • Names cannot start with the letters xml (or XML, or Xml, etc.).
- 345 • Names cannot contain spaces.
- 346 • Any name can be used; no words are reserved.

347 **3.2.2 Attribute**

348 XML elements can have attributes that provide additional information about an element. Attributes often
349 provide information that is not a part of the data. Attribute values are always quoted by using single or
350 double quotation marks.

351 **3.2.3 Substitution group**

352 A substitution group is an object-oriented feature of XSD that allows you to specify elements that can
353 replace another element in documents generated from that schema. The replaceable element is called
354 the head element and is defined in the schema's global scope. The elements of the substitution group are
355 of the same type as the head element or a type that is derived from the head element's type.

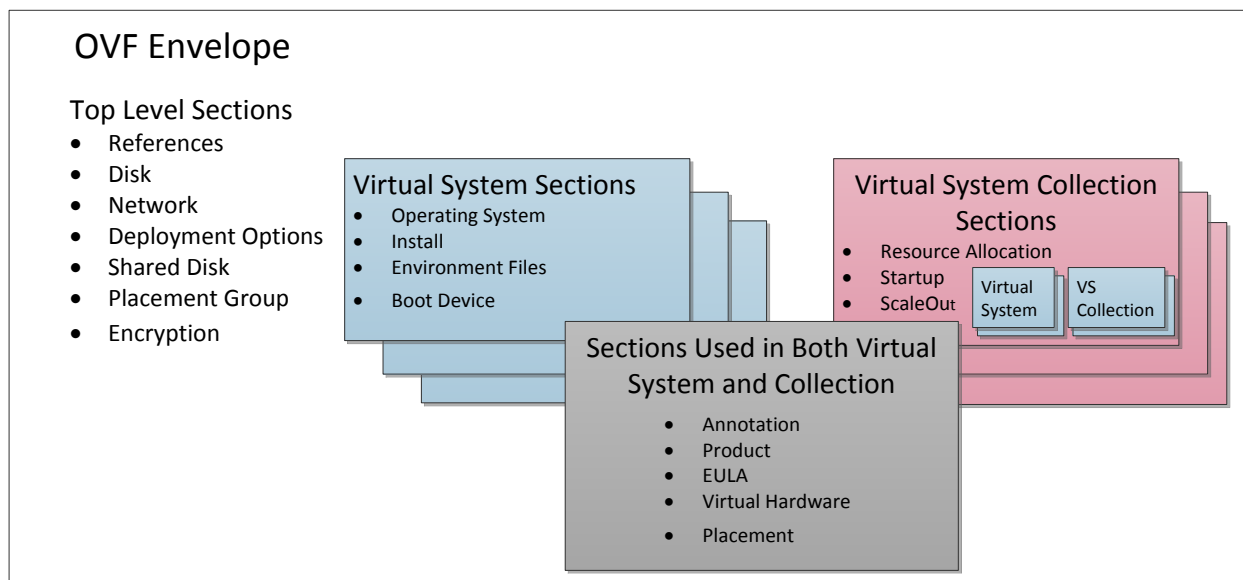
356 In essence, a substitution group allows you to build a collection of elements that can substitute for a
357 generic element. For example, if you are building an ordering system for a company that sells three types
358 of widgets, you might define a generic widget element that contains common data for all three widget
359 types. Then you can define a substitution group for the generic widget that contains more specific widget
360 types that are derived from the generic widget. In the schema, an order can be defined as a sequence of
361 generic widgets. In an XML file conformant with the schema, an order can be a sequence of widgets from
362 the substitution group rather than generic widgets. Often the head element for a substitution group is
363 declared to be abstract, so generic widgets cannot appear in the XML file.

364 4 OVF package

365 The OVF package provides a means to distribute software solutions deployed in a virtual system or
 366 collection of virtual systems. The OVF package consists of an OVF descriptor and related virtual disks.
 367 The OVF package exists as either a set of files referenced by a URL or a compressed file with the '.ova'
 368 extension.

369 4.1 Structure of an OVF descriptor

370 An OVF descriptor is a XML file. The root element of an OVF descriptor is the `Envelope`. The two most
 371 important child elements of an `Envelope` element are the `VirtualSystem` and
 372 `VirtualSystemCollection` elements. The `Envelope` element also contains sections that apply to all
 373 in `VirtualSystem` and `VirtualSystemCollection` elements the package. The `Envelope` element
 374 may contain both `VirtualSystem` and `VirtualSystemCollection` elements. The
 375 `VirtualSystemCollection` elements are a recursive construct that may, like the `Envelope` element,
 376 contain both `VirtualSystem` and `VirtualSystemCollection` elements. The OVF Schema defines
 377 a number of different sections. Some of these sections may only appear in the `Envelope` element.
 378 Others may only appear in the `VirtualSystem` element. Yet others may only appear in
 379 `VirtualSystemCollection` element. Some sections may appear in both `VirtualSystem` and
 380 `VirtualSystemCollection` elements. This structure is summarized in Figure 4.



381

382

Figure 4 – OVF package structure

383 4.2 Global attributes defined in OVF Schema

384 The following OVF attributes defined in the OVF Schema are global attributes. OVF element specific OVF
 385 attributes are also defined. (See 4.4. and 4.3.)

- 386 • `required` attribute – indicates that the deployment should fail if the element is not present or is
 387 not understood. This attribute is an XSD Boolean with allowed values are 'true', 'false', '0', '1'. If
 388 not specified, the value is 'true' or '1'. This OVF attribute should not be confused with the XSD
 389 value `required` for the XSD `use` attribute. The two terms are similar but different in
 390 significance.
- 391 • `transport` attribute – is a space-separated list of supported transport types used to convey
 392 the information to the guest software. See 4.7.3 and 6.2.

- 393 • `configuration attribute` – identifies a configuration defined in a `Configuration` element in
394 the `DeploymentOptionSection` element. See 4.4.6. The `configuration attribute` is used
395 in the following places:
 - 396 • in the `VirtualHardwareSection` elements for `Item`, `EthernetPortItem`,
397 `StorageItem` elements
 - 398 • in the `ResourceAllocationSection` element for `Item`, `EthernetPortItem`,
399 `StorageItem` elements
 - 400 • in the `ScaleOutSection` elements for `InstanceCount` elements
 - 401 • in the `ProductSection` elements for `Property` elements
- 402 • `bound attribute` – is a range marker entry used to indicate minimum, normal, and maximum
403 values for resource allocation setting data. The allowed values are 'min', 'normal', and 'max'.
404 See 4.6.1 The `bound attribute` is used in two places to set limits on resource allocation:
 - 405 • in the `VirtualHardwareSection` elements for `Item`, `EthernetPortItem`,
406 `StorageItem` elements
 - 407 • in the `ResourceAllocationSections` element for `Item`, `EthernetPortItem`,
408 `StorageItem` elements

409 4.3 Extensibility in OVF

410 The OVF Schema use the XSD elements 'any' and 'anyAttribute' to extend the OVF descriptor and
411 the OVF environment to provide custom metadata. This feature allows OVF packages to meet a wide
412 variety of use cases in the industry.

413 The following definitions come from the XML schema reference website. See
414 http://www.w3schools.com/schema/schema_elements_ref.asp.

- 415 • `any` – This definition enables the author to extend the XML document with elements not
416 specified by the schema.
- 417 • `anyAttribute` – This definition enables the author to extend the XML document with
418 attributes not specified by the schema.
- 419 • `##any` - Elements from any namespace are allowed. (This definition is the default.)
- 420 • `##other` - Elements from any namespace that is not the namespace of the parent element can
421 be present.

422 An extension at the `Envelope` element level is done by defining a new member of the `ovf:Section`
423 substitution group. An extension at the `Content` element level is done by defining a new member of the
424 `ovf:Section` substitution group. See 3.2.3. These new section elements can be used where sections are
425 allowed to be present by the OVF Schema. The `Info` element in each new section element can be used
426 to give meaningful warnings to users when a new section element is being skipped because it is not
427 understood by the deployment platform.

428 A type defined in the OVF Schema may be extended at the end with additional elements. Extension
429 points are declared with an `xs:any` with a `namespace="##other"`.

430 Additional attributes are allowed in the OVF Schema. Extension points are declared with an
431 `xs:anyAttribute`.

432 The `ovf:required` attribute specifies whether the information in the element is required or optional. The
433 `ovf:required` attribute defaults to TRUE. If the deployment platform detects an element extension that
434 is required and that it does not understand, it fails the deployment.

435 On custom attributes, the information in the attribute is not required for correct behavior.

436 EXAMPLE 1:

```
437 <!-- Optional custom section example -->
438 <otherns:IncidentTrackingSection ovf:required="false">
439   <Info>Specifies information useful for incident tracking purposes</Info>
440   <BuildSystem>Acme Corporation Official Build System</BuildSystem>
441   <BuildNumber>102876</BuildNumber>
442   <BuildDate>10-10-2008</BuildDate>
443 </otherns:IncidentTrackingSection>
```

444 EXAMPLE 2:

```
445 <!-- Open content example (extension of existing type) -->
446 <AnnotationSection>
447   <Info>Specifies an annotation for this virtual machine</Info>
448   <Annotation>This is an example of how a future element (Author) can still be
449     parsed by older clients</Annotation>
450   <!-- AnnotationSection extended with Author element -->
451   <otherns:Author ovf:required="false">John Smith</otherns:Author>
452 </AnnotationSection>
```

453 EXAMPLE 3:

```
454 <!-- Optional custom attribute example -->
455 <Network ovf:name="VM network" otherns:desiredCapacity="1 Gbit/s">
456   <Description>The main network for VMs</Description>
457 </Network>
```

458 4.4 OVF top level elements

459 The root element defined by the OVF Schema is the `Envelope` element. OVF elements that are direct
460 children of an `Envelope` element are listed below in the order that they occur in the OVF descriptor:

- 461 • `References` element
- 462 • `Section` elements - a substitution group for section elements
- 463 • `Content` elements - a substitution group for content elements
- 464 • `Strings` element

465 Elements that are a substitution group for a `Section` element that is a direct child of the `Envelope`
466 element:

- 467 • `DiskSection` element
- 468 • `NetworkSection` element
- 469 • `DeploymentOptionSection` element
- 470 • `SharedDiskSection` element
- 471 • `PlacementGroupSection` element
- 472 • `EncryptionSection` element

473 Elements that are a substitution group for a `Content` element:

- 474 • `VirtualSystem` element
- 475 • `VirtualSystemCollection` element (may be nested)

476 Elements that are a substitution group for a `Section` element used in a `Content` element are listed below
477 in the order that they occur in an OVF descriptor:

- 478 • `AnnotationSection` element
- 479 • `ProductSection` element

- 480 • OperatingSystemSection element
- 481 • EulaSection element
- 482 • VirtualHardwareSection element
- 483 • ResourceAllocationSection element
- 484 • InstallSection element
- 485 • StartupSection element
- 486 • EnvironmentFilesSection element
- 487 • BootDeviceSection element
- 488 • ScaleOutSection element
- 489 • PlacementSection element

490 The additional elements defined in the OVF Schema are listed below. Note that these elements can be
 491 used in a namespace other than `ovf:.`

- 492 • Annotation
- 493 • AppUrl
- 494 • boot:CIM_BootConfigSetting
- 495 • Category
- 496 • Configuration
- 497 • Content
- 498 • Description
- 499 • Disk
- 500 • EthernetPortItem
- 501 • File
- 502 • FullVersion
- 503 • Icon
- 504 • Info
- 505 • InstanceCount
- 506 • Item
- 507 • Label
- 508 • License
- 509 • Msg
- 510 • Name
- 511 • Network
- 512 • NetworkPortProfile
- 513 • NetworkPortProfileURI
- 514 • Product
- 515 • ProductUrl
- 516 • Property
- 517 • SharedDisk
- 518 • StorageItem
- 519 • System
- 520 • Value
- 521 • Vendor
- 522 • VendorUrl
- 523 • Version
- 524 • xenc:EncryptedKey
- 525 • xenc11:DerivedKey

526 An example of the basic structure of an OVF descriptor shown in Figure 4 is illustrated below.
 527

```
528 ovf:Envelope
529 <xs:element name="References" type="ovf:References_Type">
530
```

```

531 <xs:element ref="ovf:Section" minOccurs="0" maxOccurs="unbounded">
532 <DiskSection>
533   <Info> Describes all virtual disks used with the package </Info>
534 <NetworkSection>
535   <Info>List of logical networks used in the package</Info>
536 <DeploymentOptionSection>
537   <Info>List of deployment options available in the package</Info>
538
539 <xs:element ref="ovf:Content">
540   <VirtualSystemCollection ovf:id="Acme VSC">
541     <Info>The packaging of the first virtual appliance</Info>
542     -----
543     <VirtualSystem ovf:id="Acme VS 1">
544       <Info>The packaging of the virtual machine 1</Info>
545     <VirtualSystem ovf:id="Acme VS 2">
546       <Info>The packaging of the virtual machine 2</Info>
547     -----
548     <VirtualSystemCollection ovf:id="Widget VSC">
549       <Info>The packaging of the second virtual appliance</Info>
550     <VirtualSystem ovf:id="Acme VS 3">
551       <Info>The packaging of the virtual machine 3</Info>
552     <VirtualSystem ovf:id="Acme VS 3">
553       <Info>The packaging of the virtual machine 4</Info>
554     -----
555 <xs:element name="Strings" type="ovf:Strings_Type" minOccurs="0"
556 maxOccurs="unbounded">
557   <Info> Root element of I18N string bundle</Info>

```

558 4.4.1 VirtualSystem element

559 A `VirtualSystem` element is a substitution element for a `Content` element. It contains a number of
560 `Section` elements that define a single virtual system. These section elements describe virtual hardware,
561 the resource allocation, and product information applicable to the virtual system.

562 4.4.2 VirtualSystemCollection element

563 A `VirtualSystemCollection` element is a substitution element for a `Content` element. It contains
564 one or more `VirtualSystem` elements and a number of `Section` elements that define a collection of
565 virtual systems. These section elements describe virtual hardware, the resource allocation, and product
566 information applicable to the virtual system collection.

567 4.4.3 References element

568 The `References` element contains the references to all external files.

569 4.4.4 DiskSection element

570 The `DiskSection` element defines the virtual disks used by the virtual systems in the OVF package.

571 Any virtual disk format may be used, as long as the virtual disk format specification is public and available
572 without restrictions. This supports the full range of virtual hard disk formats used for hypervisors today,
573 and it is extensible to allow for future formats.

574 The virtual disk format may be a simple, basic disk block format agnostic to the guest software installed.
 575 For example, VMware VMDK formats deal with 512-byte disk sectors stored in 64KB blocks, in a number
 576 of flat, sparse, and compressed variants. At deployment time, the virtualization platform creates virtual
 577 disks in a basic disk block format it prefers. The run-time virtual disk format may be identical to the
 578 distribution format, but is often different because it may not be efficient to run out of a compressed virtual
 579 disk format. The guest software that is installed has its own file system format, e.g., NTFS, EXT3, or ZFS.
 580 The OVF virtual disk though does not need to know the file system format.

581 The following example shows a description of virtual disks:

```
582 <DiskSection>
583   <Info>Describes the set of virtual disks</Info>
584   <Disk ovf:diskId="vmdisk1" ovf:fileRef="file1" ovf:capacity="8589934592"
585     ovf:populatedSize="3549324972"
586     ovf:format=
587       "http://www.vmware.com/interfaces/specifications/vmdk.html#sparse">
588   </Disk>
589   <Disk ovf:diskId="vmdisk2" ovf:capacity="536870912"
590   </Disk>
591   <Disk ovf:diskId="vmdisk3" ovf:capacity="${disk.size}"
592     ovf:capacityAllocationUnits="byte * 2^30"
593   </Disk>
594 </DiskSection>
```

595 4.4.5 NetworkSection element

596 The `NetworkSection` element describes the network and network connections used when the OVF
 597 package is deployed. The network may be defined in simplistic terms, e.g., as a Red, Green, Blue
 598 network. The Ethernet Port characteristics are defined by the
 599 `CIM_EthernetPortAllocationSettingData` class. It may also be defined in greater detail with the
 600 use of the *Network Port Profile* (DSP8049), which may be included in an OVF package.

601 There is a basic assumption that a flat layer 2 network is available for which the Ethernet ports to
 602 connect. There is no assumption made regarding the services or characteristics of that network. The
 603 speed of the Ethernet port can be set by using the `CIM_EthernetPortAllocationSettingData`.

604 4.4.5.1 OVF networks

605 The following example is a snippet of an OVF descriptor for the Network Section. It simply states that the
 606 Ethernet ports defined in the virtual systems are attached to the “Red” network. The `epasd:Connection`
 607 property specifies the network to which the Ethernet port is connected. The simplistic assumption is that
 608 all Ethernet ports are connected to a layer 2 network. If more than one network is defined, e.g., “Red” and
 609 “Green”, there is an assumption that the “Red” and “Green” networks are not connected. A connection to
 610 an external network, e.g., a data center LAN or a WAN, is considered part of the deployment provisioning
 611 and not specified in the OVF descriptor. In this case the agreement between the consumer of the OVF
 612 package and the hosting service determines the network characteristics and services.

```

613 <!-- Describes all networks used in the package -->
614 <NetworkSection>
615   <Info>Logical networks used in the package</Info>
616   <Network ovf:name="Red Network">
617     <Description>The network that the virtual systems are attached to.
618     on</Description>
619   </Network>
620 </NetworkSection>
621

```

622 The following example is a snippet of an item in an OVF descriptor `VirtualHardwareSection` in a
623 `VirtualSystem` element that illustrates how the virtual system is attached to the “Red” network by using
624 the `Connection` property from the `CIM_EthernetPortAllocationSettingData` class.

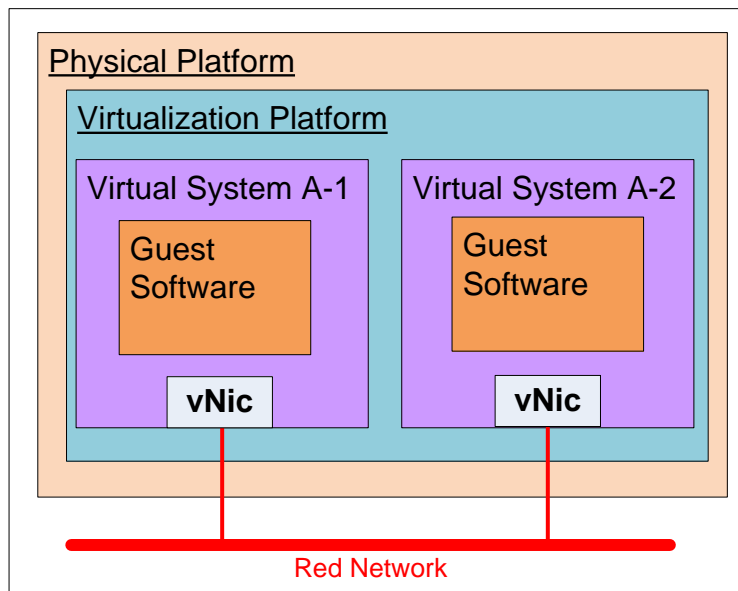
```

625 <EthernetPortItem>
626   <epasd:AddressOnParent>7</epasd:AddressOnParent>
627   <epasd:AutomaticAllocation>true</epasd:AutomaticAllocation>
628   <epasd:Connection>Red Network</epasd:Connection>
629   <epasd:Description>Virtual Ethernet adapter</epasd:Description>
630   <epasd:ElementName>Virtual NIC 1</epasd:ElementName>
631   <epasd:InstanceID>8</epasd:InstanceID>
632   <epasd:ResourceType>10</epasd:ResourceType>
633 </EthernetPortItem>

```

634 More than one network may be defined.

635 Figure 5 illustrates a simplistic network. The Ethernet port connects to the “Red” network.

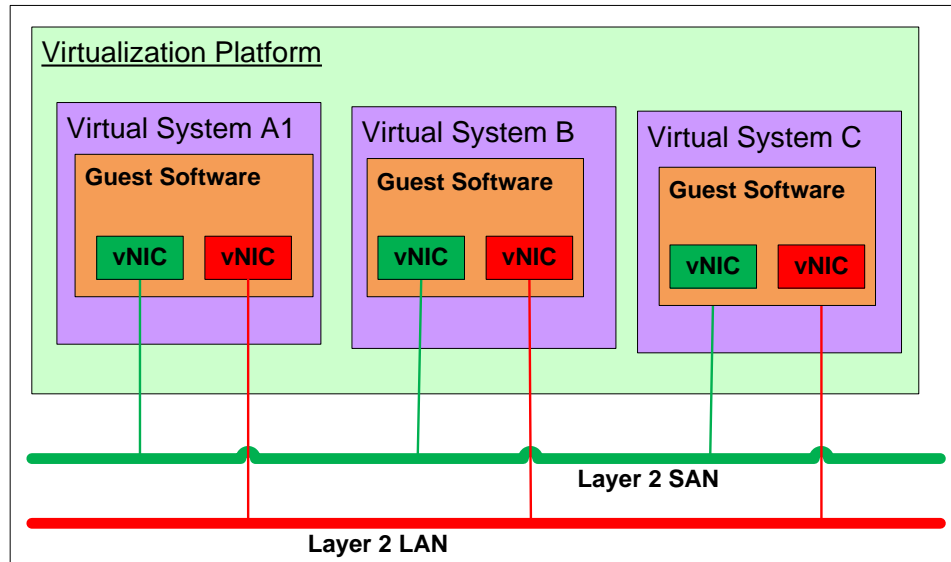


636

637

Figure 5 – Network connections

638 Figure 6 illustrates a dual network configuration: a “Green” network for storage connections and a “Red”
639 network for local area network connections.



640

641

Figure 6 – LAN-SAN network connections

642 **4.4.5.2 Network Port Profile**

643 The *Network Port Profile* (DSP8049) defines a configuration of the properties of a network's network ports
 644 that is used for communication by the virtual systems defined in the OVF package. See the *Virtual*
 645 *Networking Management White Paper* (DSP2025) for additional information.

646 A complete listing of the properties in the `EthernetPortAllocationSettingData` class that can be
 647 specified is shown below. The allowed values for each property are defined in the DSP0243, DSP1041,
 648 DSP1050, and the CIM Schema. Only a subset of these properties is used in most OVF descriptors.

```

649 <EthernetPortItem>
650   <epasd:Address>
651   <epasd:AddressOnParent>
652   <epasd:AllocationUnits>
653   <epasd:AllowedPriorities>
654   <epasd:AllowedToReceiveMACAddresses>
655   <epasd:AllowedToReceiveVLANs>
656   <epasd:AllowedToTransmitMACAddresses>
657   <epasd:AllowedToTransmitVLANs>
658   <epasd:AutomaticAllocation>
659   <epasd:AutomaticDeallocation>
660   <epasd:Caption>
661   <epasd:ChangeableType>
662   <epasd:ConfigurationName>
663   <epasd:Connection>
664   <epasd:ConsumerVisibility>
665   <epasd:DefaultPortVID>
666   <epasd:DefaultPriority>
667   <epasd:Description>
668   <epasd:DesiredVLANEndpointMode>
669   <epasd:ElementName>
670   <epasd:GroupID>
    
```

```

671 <epasd:HostResource>
672 <epasd:InstanceID>
673 <epasd:Limit>
674 <epasd:ManagerID>
675 <epasd:MappingBehavior>
676 <epasd:NetworkPortProfileID>
677 <epasd:NetworkPortProfileIDType>
678 <epasd:OtherEndpointMode>
679 <epasd:OtherNetworkPortProfileIDTypeInfo>
680 <epasd:OtherResourceType>
681 <epasd:Parent>
682 <epasd:PoolID>
683 <epasd:PortCorrelationID>
684 <epasd:PortVID>
685 <epasd:Promiscuous>
686 <epasd:ReceiveBandwidthLimit>
687 <epasd:ReceiveBandwidthReservation>
688 <epasd:Reservation>
689 <epasd:ResourceSubType>
690 <epasd:ResourceType>
691 <epasd:SourceMACFilteringEnabled>
692 <epasd:VSITypeID>
693 <epasd:VSITypeIDVersion>
694 <epasd:VirtualQuantity>
695 <epasd:VirtualQuantityUnits>
696 <epasd:Weight>
697 </EthernetPortItem>

```

698 4.4.6 DeploymentOptionsSection element

699 The `DeploymentOptionsSection` element is a direct child element of the `Envelope` element. The
700 `Configuration` element is a direct child element of the `DeploymentOptionsSection` element. A
701 `DeploymentOptionsSection` element contains one or more `Configuration` elements.

702 The `Configuration` element is used to specify a resource configuration used when an OVF package is
703 deployed. A choice of one of the configurations is made at deployment. The user input can be requested
704 through the use of the `userConfigurable` attribute.

705 The `DeploymentOptionsSection` element lists the IDs, labels and descriptions of the configurations
706 available in the OVF package.

707 A default configuration is indicated by setting the `default` attribute to `true`. In the absence of other
708 input, the default configuration is used. If a default is not indicated, the first configuration is taken as the
709 default.

710 Each configuration has a unique `ID` attribute that identifies that configuration. This value of the `ID`
711 attribute is specified in a `configuration` attribute in other section elements, such as a
712 `VirtualHardwareSection` element, a `ProductSection` element, or a `ScaleOutSection`
713 element.

714 If an element is to appear in more than one configuration, the `configuration` attribute of the elements
715 is a list of space-separated configuration IDs.

716 The deployment function needs to select a configuration either from user input or from other metadata.
 717 When a configuration is selected, elements with a `configuration` attribute set to the selected
 718 configuration are used for the deployment. Elements without a `configuration` attribute are also
 719 deployed, but those with a different configuration attribute are not deployed. Thus, selecting a single
 720 configuration during deployment may affect the configuration of many different items in different sections
 721 of the OVF package.

722 A snippet from an OVF descriptor `DeploymentOptionsSection` element that illustrates the use of the
 723 `Configuration` element is shown below. Note that the default configuration for the memory resource
 724 allocation is highlighted in blue and the “big” configuration is highlighted in gray. The deployment function
 725 chooses the default configuration unless there is consumer input to choose the “big” configuration.
 726

```

727 <DeploymentOptionsSection>
728   <Configuration ovf:id="big">
729     <Label>Big</Label>
730     <Description>Apply reservations for Memory</Description>
731   </Configuration>
732   ...
733 </DeploymentOptionsSection>
734
735 <VirtualHardwareSection>
736   <Info>...</Info>
737
738   <Item>
739     <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
740     <rasd:ElementName>512 MB memory size-no reservation</rasd:ElementName>
741     <rasd:InstanceID>0</rasd:InstanceID>
742     <rasd:ResourceType>4</rasd:ResourceType>
743     <rasd:VirtualQuantity>512</rasd:VirtualQuantity>
744   </Item>
745
746   <Item ovf:configuration="big">
747     <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
748     <rasd:ElementName>512 MB memory size &256 MB
749     reservation</rasd:ElementName>
750     <rasd:InstanceID>0</rasd:InstanceID>
751     <rasd:Reservation>256</rasd:Reservation>
752   </Item>
753 </VirtualHardwareSection>
  
```

754
 755 The resulting CIM instance for the resource allocation is illustrated below.

```

756 <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
757 <rasd:ElementName>512 MB memory size & 256 MB reservation </rasd:ElementName>
758 <rasd:InstanceID>0</rasd:InstanceID>
759 <rasd:Reservation>256</rasd:Reservation>
760 <rasd:ResourceType>4</rasd:ResourceType>
761 <rasd:VirtualQuantity>512</rasd:VirtualQuantity>
  
```

762 The text highlighted in gray is from the “big” configuration. The text highlighted in blue is from the default
 763 configuration.

764 The following example illustrates a `DeploymentOptionsSection` element with three `Configuration`
 765 elements. Note that the “normal” configuration is designated as the default, so if no configuration is
 766 selected, the deployment uses the elements with a `configuration` attribute of “normal”.

```

767 <DeploymentOptionSection>
768   <Configuration ovf:id="minimal">
769     <Label>Minimal</Label>
770     <Description>Smallest practical implementation</Description>
771   </Configuration>
772   <Configuration ovf:id="normal" ovf:default="true">
773     <Label>Normal</Label>
774     <Description>A typical implementation</Description>
775   </Configuration>
776   <Configuration ovf:id="large">
777     <Label>Large</Label>
778     <Description>A scaled up implementation</Description>
779   </Configuration>
780 </DeploymentOptionSection>

```

781 The following example illustrates the use of the `configuration` attribute to define the resource
782 allocations for the three configurations above. The `VirtualHardwareSection` element uses the
783 `CIM_ResourceAllocationDespcriptor` properties to specify the desired memory resource
784 configuration. Each `Item` element in the `VirtualHardwareSection` refers to a configuration defined
785 in the `DeploymentOptionsSection` element above.

```

786 <VirtualHardwareSection>
787   <Info>...</Info>
788   <Item ovf:configuration="normal">
789     <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
790     <rasd:ElementName>512 MB memory size and 256 MB reservation</rasd:ElementName>
791     <rasd:InstanceID>0</rasd:InstanceID>
792     <rasd:Reservation>256</rasd:Reservation>
793     <rasd:ResourceType>4</rasd:ResourceType>
794     <rasd:VirtualQuantity>512</rasd:VirtualQuantity>
795   </Item>
796   <Item ovf:configuration="minimal">
797     <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
798     <rasd:ElementName>256 MB memory size and 128 MB reservation</rasd:ElementName>
799     <rasd:InstanceID>1</rasd:InstanceID>
800     <rasd:Reservation>128</rasd:Reservation>
801     <rasd:ResourceType>4</rasd:ResourceType>
802     <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
803   </Item>
804   <Item ovf:configuration="large">
805     <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
806     <rasd:ElementName>1024 MB memory size and 512 MB reservation</rasd:ElementName>
807     <rasd:InstanceID>0</rasd:InstanceID>
808     <rasd:Reservation>512</rasd:Reservation>
809     <rasd:ResourceType>4</rasd:ResourceType>
810     <rasd:VirtualQuantity>1024</rasd:VirtualQuantity>
811   </Item>
812 </VirtualHardwareSection>

```

813 In the example above, the memory size is controlled by the configuration selected during deployment. If a
814 configuration is not selected, the memory size defaults to the "normal" configuration.

815 The following example shows the use of a configuration attribute in a `ProductSection`.

```
816 <ProductSection>
817   <Property ovf:key="app_log" ovf:type="string" ovf:value="low"
818     ovf:configuration="normal">
819     <Label>Loglevel</Label>
820     <Description>Loglevel for the service</Description>
821     <Value ovf:value="none" ovf:configuration="minimal">
822     <Value ovf:value="high" ovf:configuration="large">
823   </Property>
824 </ProductSection>
```

825 In this example, the value of “app_log” changes based on the configuration selected. As with virtual
826 hardware, if a configuration is not selected, the value of the “app_log” property defaults to the “normal”
827 configuration value, i.e., “low”.

828 4.4.7 SharedDiskSection element

829 The `SharedDiskSection` element allows a virtual disk to be referenced by multiple virtual systems to
830 satisfy the needs of clustered databases. The file sharing system technology used is platform specific.

831 The `SharedDiskSection` element is a valid only at the envelope level.

832 Each shared disk has a unique identifier for the OVF package. The `SharedDiskSection` element adds
833 a Boolean `ovf:readOnly` attribute that indicates whether read-write (i.e., `FALSE`), or read-only (i.e.,
834 `TRUE`) access is allowed.

835 The following example illustrates the basics of a `SharedDiskSection` element.

```
836 <ovf:SharedDiskSection>
837   <Info>Describes the set of virtual disks shared between VMs</Info>
838   <ovf:SharedDisk ovf:diskId="datadisk" ovf:fileRef="data"
839     ovf:capacity="8589934592" ovf:populatedSize="3549324972"
840     ovf:format="http://www.vmware.com/interfaces/specifications/vmdk.html#sparse"/>
841   <ovf:SharedDisk ovf:diskId="transientdisk" ovf:capacity="536870912"/>
842 </ovf:SharedDiskSection>
```

843 The following example illustrates the use of shared disks. The disks for installations of the operating
844 system (system), the cluster software (crs_home), and database (db_home) are backed by external `File`
845 references. The shared virtual disks in this example have no backing by an external `File` reference; the
846 deployment engine creates the shared disk appropriately to be shared by more than one virtual system.

```
847 <ovf:References>
848   <ovf:File ovf:id="system" ovf:href="system.img" ovf:compression="gzip"/>
849   <ovf:File ovf:id="crs_home" ovf:href="crs_home.img" ovf:compression="gzip"/>
850   <ovf:File ovf:id="db_home" ovf:href="db_home.img" ovf:compression="gzip"/>
851 </ovf:References>
852 <ovf:DiskSection>
853   <ovf:Info>Virtual Disks</ovf:Info>
854   <ovf:Disk ovf:diskId="system" ovf:fileRef="system" ovf:capacity="5368709120"
855   ovf:format="Raw disk image"/>
856   <ovf:Disk ovf:diskId="crs_home" ovf:fileRef="crs_home" ovf:capacity="2147483648"
857   ovf:format="Raw disk image"/>
858   <ovf:Disk ovf:diskId="db_home" ovf:fileRef="db_home" ovf:capacity="4294967296"
859   ovf:format="Raw disk image"/>
```

```

860 </ovf:DiskSection>
861 <ovf:SharedDiskSection>
862   <ovf:Info>Virtual Disks shared at runtime</ovf:Info>
863   <ovf:SharedDisk ovf:diskId="crs_asm" ovf:capacity="4294967296" ovf:format="Raw disk
864 image"/>
865   <ovf:SharedDisk ovf:diskId="db_asm" ovf:capacity="12884901888" ovf:format="Raw disk
866 image"/>
867 </ovf:SharedDiskSection>
868 .....
869 <ovf:VirtualSystemCollection ovf:id="rac_db_asm">
870   <ovf:Info>Sample Oracle RAC using ASM</ovf:Info>
871   .....
872   <ovf:ScaleOutSection ovf:id="rac_db">
873     <ovf:Info>RAC DB</ovf:Info>
874     <ovf:Description>Number of instances</ovf:Description>
875     <ovf:InstanceCount ovf:default="2" ovf:minimum="2"
876 ovf:maximum="4"></ovf:InstanceCount>
877   </ovf:ScaleOutSection>
878   .....
879   <ovf:VirtualSystem ovf:id="rac_db">
880     <ovf:Info>RAC DB Instance</ovf:Info>
881     .....
882     <ovf:VirtualHardwareSection>
883       <ovf:Info>System requirements: 8192 MB, 2 CPUs, 5 disks, 2 nics
884 </ovf:Info>
885       .....
886       <ovf:Item>
887         <rasd:Description>Disk 1</rasd:Description>
888         <rasd:ElementName>Disk 1</rasd:ElementName>
889         <rasd:HostResource>ovf:/disk/system</rasd:HostResource>
890         <rasd:ResourceType>17</rasd:ResourceType>
891       </ovf:Item>
892       <ovf:Item>
893         <rasd:Description>Disk 2</rasd:Description>
894         <rasd:ElementName>Disk 2</rasd:ElementName>
895         <rasd:HostResource>ovf:/disk/crs_home</rasd:HostResource>
896         <rasd:ResourceType>17</rasd:ResourceType>
897       </ovf:Item>
898       <ovf:Item>
899         <rasd:Description>Disk 3</rasd:Description>
900         <rasd:ElementName>Disk 3</rasd:ElementName>
901         <rasd:HostResource>ovf:/disk/db_home</rasd:HostResource>
902         <rasd:ResourceType>17</rasd:ResourceType>
903       </ovf:Item>
904       <ovf:Item>
905         <rasd:Description>Disk 4</rasd:Description>
906         <rasd:ElementName>Disk 4</rasd:ElementName>
907         <rasd:HostResource>ovf:/disk/crs_asm</rasd:HostResource>
908         <rasd:ResourceType>17</rasd:ResourceType>
909       </ovf:Item>

```

```

910     <ovf:Item>
911         <rasd:Description>Disk 5</rasd:Description>
912         <rasd:ElementName>Disk 5</rasd:ElementName>
913         <rasd:HostResource>ovf:/disk/db_asm</rasd:HostResource>
914         <rasd:ResourceType>17</rasd:ResourceType>
915     </ovf:Item>
916     .....
917 </ovf:VirtualHardwareSection>
918 </ovf:VirtualSystem>
919 </ovf:VirtualSystemCollection>

```

920 4.4.8 PlacementGroupSection element

921 The PlacementGroupSection element defines an ID for a placement group and its associated
 922 placement policy(s). The placement group is associated with a VirtualSystemCollection or
 923 VirtualSystem through the use of the PlacementSection element.

924 An example of PlacementGroupSection elements is shown below.

```

925     <ovf:PlacementGroupSection ovf:id="PG2" ovf:policy="availability">
926         <Info>Placement policy for group of virtual systems that need availability</Info>
927         <ovf:Description>Placement policy for a database tier</ovf:Description>
928     </ovf:PlacementGroupSection>
929     ...
930     <ovf:PlacementGroupSection ovf:id="PG1" ovf:policy="affinity">
931         <Info>Placement policy for group of virtual systems that need affinity</Info>
932         <ovf:Description>Placement policy for a web tier</ovf:Description>
933     </ovf:PlacementGroupSection>

```

934 The PlacementGroupSection element is a direct child element of the Envelope element. See 4.5.5.

935 4.5 OVF section elements used in Virtual System and Virtual System Collection

936 The following OVF descriptor section elements may appear within a VirtualSystem or
 937 VirtualSystemCollection element.

938 4.5.1 AnnotationSection element

939 The AnnotationSection element is user-defined and can appear in VirtualSystem and
 940 VirtualSystemCollection elements. An AnnotationSection element contains one Annotation
 941 element. Annotation elements are localizable. A suggested use for Annotation elements is to display
 942 them to the consumers as the package is deployed. Note that the Annotation element specified in OVF
 943 is not an XML Schema Annotation element.

```

944 <AnnotationSection>
945     <Info>An annotation on this service. It can be ignored</Info>
946     <Annotation>Contact customer support if you have any problems</Annotation>
947 </AnnotationSection >

```

948 4.5.2 ProductSection element

949 The ProductSection element provides product information such as name and vendor of the appliance
 950 and a set of properties that can be used to customize the appliance. These properties are be configured

951 at installation time of the appliance, typically by prompting the user. This is discussed in more detail
 952 below.

```

953 <ProductSection ovf:class="com.mycrm.myservice" ovf:instance="1">
954   <Info>Describes product information for the service</Info>
955   <Product>MyCRM Enterprise</Product>
956   <Vendor>MyCRM Corporation</Vendor>
957   <Version>4.5</Version>
958   <FullVersion>4.5-b4523</FullVersion>
959   <ProductUrl>http://www.mycrm.com/enterprise</ProductUrl>
960   <VendorUrl>http://www.mycrm.com</VendorUrl>
961   <Icon ovf:height="32" ovf:width="32" ovf:mimeType="image/png" ovf:fileRef="icon">
962   <Category>Email properties</Category>
963   <Property ovf:key="adminEmail" ovf:type="string" ovf:userConfigurable="true">
964     <Label>Admin email</Label>
965     <Description>Email address of administrator</Description>
966   </Property>
967   <Category>Admin properties</Category>
968   <Property ovf:key="appLog" ovf:type="string" ovf:value="low"
969   ovf:userConfigurable="true">
970     <Description>Loglevel for the service</Description>
971   </Property>
972   <Property ovf:key="appisSecondary" ovf:value="false" ovf:type="boolean">
973     <Description>Cluster setup for application server</Description>
974   </Property>
975   <Property ovf:key="appIp" ovf:type="string" ovf:value="{appserver-vm}">
976     <Description>IP address of the application server VM</Description>
977   </Property>
978 </ProductSection>
  
```

979 Note that the `ovf:key` attribute does not contain the period character ('.') or the colon character (':') and
 980 the `ovf:class` and `ovf:instance` attributes do not contain the colon character (':').

981 If only one instance of a product is installed, the `ovf:instance` attribute is not used.

982 The following illustrates the use of OVF Properties in a `ProductSection` element:

```

983 <ProductSection>
984   <Property ovf:key="adminEmail" ovf:type="string" ovf:userConfigurable="true"
985   ovf:configuration="standard">
986     <Label>Admin email</Label>
987     <Description>Email address of service administrator</Description>
988   </Property>
989   <Property ovf:key="appLog" ovf:type="string" ovf:value="low"
990   ovf:userConfigurable="true">
991     <Label>Loglevel</Label>
992     <Description>Loglevel for the service</Description>
993     <Value ovf:value="none" ovf:configuration="minimal">
994   </Property>
995 </ProductSection>
  
```

996 In the example above, the `adminEmail` property is only user configurable in the standard configuration,
 997 while the default value for the `appLog` property is changed from “low” to “none” in the minimal
 998 configuration.

999 4.5.3 EulaSection element

1000 The `EulaSection` contains the human readable licensing agreement for its parent, usually a
 1001 `VirtualSystem` or `VirtualSystemCollection` element. Each parent element may have more than
 1002 one `EulaSection` element. The contents of the `License` element of each `EulaSection` element
 1003 are displayed to the user for acceptance when the OVF package is deployed. If unattended deployment is
 1004 supported, provision is made for implicit acceptance of the Eula.

1005 Eulas may be externalized for localization or to point to an external license document. See 5.2 for more
 1006 details about internationalization.

1007 This is an example `EulaSection` element:

```
1008 <EulaSection>
1009     <Info>Licensing agreement</Info>
1010     <License>
1011     Lorem ipsum dolor sit amet, ligula suspendisse nulla pretium, rhoncus tempor placerat
1012     fermentum, enim integer ad vestibulum volutpat. Nisl rhoncus turpis est, vel elit,
1013     congue wisi enim nunc ultricies sit, magna tincidunt. Maecenas aliquam maecenas ligula
1014     nostra, accumsan taciti. Sociis mauris in integer, a dolor netus non dui aliquet,
1015     sagittis felis sodales, dolor sociis mauris, vel eu libero cras. Interdum at. Eget
1016     habitasse elementum est, ipsum purus pede porttitor class, ut adipiscing, aliquet sed
1017     auctor, imperdiet arcu per diam dapibus libero duis. Enim eros in vel, volutpat nec
1018     pellentesque leo, scelerisque.
1019     </License>
1020 </EulaSection>
```

1021 4.5.4 VirtualHardwareSection element

1022 The `VirtualHardwareSection` element describes the virtual hardware that is using the CIM resource
 1023 allocation setting data model. This model is based on `CIM_ResourceAllocationSettingData`
 1024 classes that specify CIM properties to describe the type and quantity of the resource being requested.
 1025 The CIM Schema is available at <http://www.dmtf.org/standards/cim>.

1026 The minimum required resource allocation setting data for a virtual processor device is illustrated below.
 1027 This is not user friendly, so it helps to add the `rasd:Description` and `rasd:ElementName`.

```
1028 <Item>
1029     <rasd:InstanceID>0</epasd:InstanceID>
1030     <rasd:ResourceType>3</epasd:ResourceType>
1031 </Item>
```

1032 This `VirtualHardwareSection` element describes the virtual devices in the hardware abstraction
 1033 layer that is used by a virtual system. The `CIM_ResourceAllocationSettingData` has a list of
 1034 devices. Some devices, such as the Ethernet port and Storage, are subclassed with an extended set of
 1035 properties.

1036 In this particular case, a fairly typical set of hardware (500 MB of guest memory, 1 CPU, 1 NIC, and one
 1037 virtual disk) is specified. The network and disk identifiers from the outer sections are referenced here. An
 1038 incomplete or missing hardware section may cause the deployment to fail.

1039 The following illustrates a `VirtualHardwareSection` element.

```

1040 <VirtualHardwareSection>
1041   <Info>Memory = 4 GB, CPU = 1 GHz, Disk = 100 GB, 1 Ethernet nic</Info>
1042   <Item>
1043     <rasd:AllocationUnits>Hertz*10^9</rasd:AllocationUnits>
1044     <rasd:Description>Virtual CPU</rasd:Description>
1045     <rasd:ElementName>1 GHz virtual CPU</rasd:ElementName>
1046     <rasd:InstanceID>1</rasd:InstanceID>
1047     <rasd:Reservation>1</rasd:Reservation>
1048     <rasd:ResourceType>3</rasd:ResourceType>
1049     <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
1050     <rasd:VirtualQuantityUnit>Count</ rasd:VirtualQuantityUnit>
1051   </Item>
1052   <Item>
1053     <rasd:AllocationUnits>byte*2^30</rasd:AllocationUnits>
1054     <rasd:Description>Memory</rasd:Description>
1055     <rasd:ElementName>1 GByte of memory</rasd:ElementName>
1056     <rasd:InstanceID>2</rasd:InstanceID>
1057     <rasd:Limit>4</rasd:Limit>
1058     <rasd:Reservation>41</rasd:Reservation>
1059     <rasd:ResourceType>4</rasd:ResourceType>
1060   </Item>
1061   <EthernetPortItem>
1062     <epasd:AllocationUnits>bit / second *2^30 </rasd:AllocationUnits>
1063     <epasd:Connection>VM Network</epasd:Connection>
1064     <epasd:Description>Virtual NIC</epasd:Description>
1065     <epasd:ElementName>Ethernet Port</epasd:ElementName>
1066     <epasd:NetworkPortProfileID>1</epasd:NetworkPortProfileID>
1067     <epasd:NetworkPortProfileIDType>4</epasd:NetworkPortProfileIDType>
1068     <epasd:ResourceType>10</epasd:ResourceType>
1069     <epasd:VirtualQuantity>1</epasd:VirtualQuantity>
1070     <epasd:VirtualQuantityUnits>Count</epasd:VirtualQuantityUnits>
1071   </EthernetPortItem>
1072   <StorageItem>
1073     <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
1074     <sasd:Description>Virtual Disk</sasd:Description>
1075     <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
1076
1077     <sasd:Reservation>100</sasd:Reservation>
1078     <sasd:ResourceType>31</sasd:ResourceType>
1079     <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
1080     <sasd:VirtualQuantityUnit>Count</sasd:VirtualQuantityUnit>
1081   </StorageItem>
1082 </VirtualHardwareSection>

```

1083 An example of a ResourceSubType CIM property follows:

```

1084   <rasd:ResourceSubType>buslogic lsilogic</rasd:ResourceSubType>

```

1085 The following example illustrates a VirtualHardwareSection element with a default and a 'big'
1086 configuration. See 4.4.6 for information about how configuration options are used.

```

1087 <VirtualHardwareSection>
1088   <Info>...</Info>
1089   <Item>
1090     <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1091     <rasd:ElementName>512 MB memory size and 256 MB reservation</rasd:ElementName>
1092     <rasd:InstanceID>0</rasd:InstanceID>
1093     <rasd:Reservation>256</rasd:Reservation>
1094     <rasd:ResourceType>4</rasd:ResourceType>
1095     <rasd:VirtualQuantity>512</rasd:VirtualQuantity>
1096   </Item>
1097   ...
1098   <Item ovf:configuration="big">
1099     <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1100     <rasd:ElementName>1024 MB memory size and 512 MB reservation</rasd:ElementName>
1101     <rasd:InstanceID>0</rasd:InstanceID>
1102     <rasd:Reservation>512</rasd:Reservation>
1103     <rasd:ResourceType>4</rasd:ResourceType>
1104     <rasd:VirtualQuantity>1024</rasd:VirtualQuantity>
1105   </Item>
1106 </VirtualHardwareSection>

```

1107 4.5.5 PlacementSection element

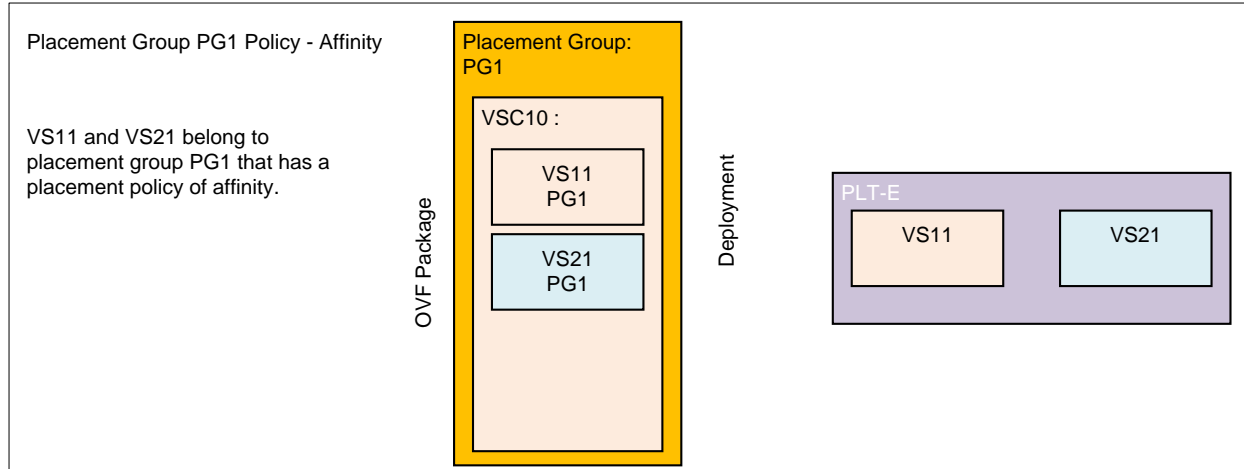
1108 The PlacementSection element specifies the placement group of which a Virtual System or Virtual
 1109 System Collection is a member. The placement policy(s) specified in the placement group (see 4.4.8) is
 1110 applied by the deployment function. The following OVF descriptor snippet illustrates a placement section
 1111 in each of two virtual systems.

```

1112 <VirtualSystemCollection ovf:id="VSC10">
1113   <VirtualSystem ovf:id="VS11">
1114     <Info>Web server</Info>
1115     ...
1116     <ovf:PlacementSection ovf:group="PG1">
1117       <Info>Placement policy group reference</Info>
1118     </ovf:PlacementSection>
1119     ...
1120   </VirtualSystem>
1121   <VirtualSystem ovf:id="VS21">
1122     <Info>Web server</Info>
1123     ...
1124     <ovf:PlacementSection ovf:group="PG1">
1125       <Info>Placement policy group reference</Info>
1126     </ovf:PlacementSection>
1127     ...
1128   </VirtualSystem>
1129 </VirtualSystemCollection>

```

1130 In this example, the virtual systems, when instantiated, should be placed according to the placement
 1131 policies specified in the "PG1" placement group. As shown in 4.4.8, the placement group 'PG1' has an
 1132 affinity placement policy. Figure 7 illustrates an affinity placement.



1133

1134

Figure 7 – Affinity placement

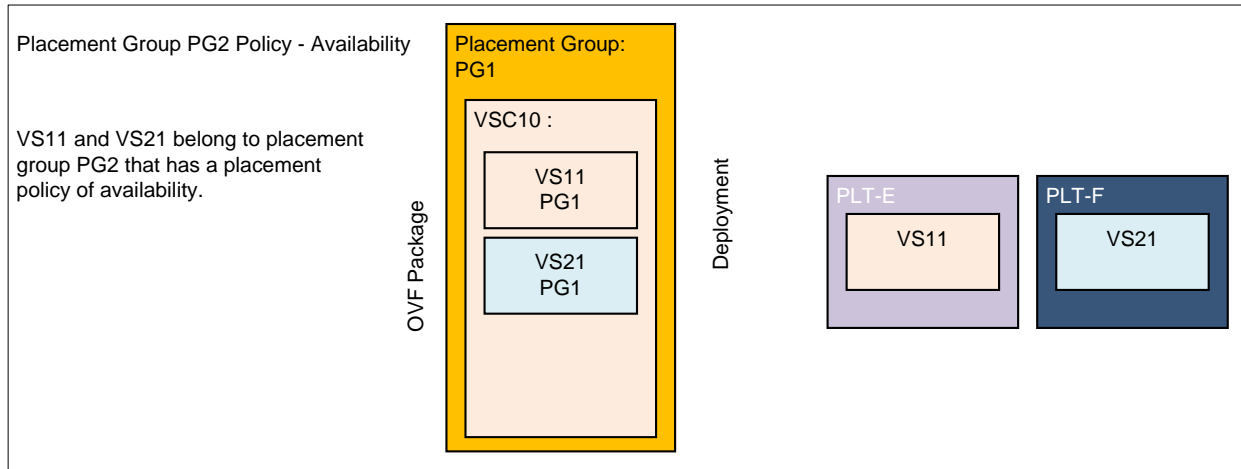
1135 The following OVF descriptor snippet illustrates a placement section in each of two virtual systems.

```

1136 <VirtualSystemCollection ovf:id="VSC10">
1137   <VirtualSystem ovf:id="VS11">
1138     <Info>Web server</Info>
1139     ...
1140     <ovf:PlacementSection ovf:group="PG2">
1141       <Info>Placement policy group reference</Info>
1142     </ovf:PlacementSection>
1143     ...
1144   </VirtualSystem>
1145   <VirtualSystem ovf:id="VS21">
1146     <Info>Web server</Info>
1147     ...
1148     <ovf:PlacementSection ovf:group="PG2">
1149       <Info>Placement policy group reference</Info>
1150     </ovf:PlacementSection>
1151     ...
1152   </VirtualSystem>
1153 </VirtualSystemCollection>

```

1154 In this example, the virtual systems, when instantiated, should be placed according to the placement
 1155 policies specified in the "PG2" placement group. As shown in 4.4.8, the placement group 'PG2' has an
 1156 availability placement policy. Figure 8 illustrates an availability placement.



1157

1158

Figure 8 – Availability placement

1159 The following OVF descriptor snippet illustrates a placement section in each of two virtual systems.

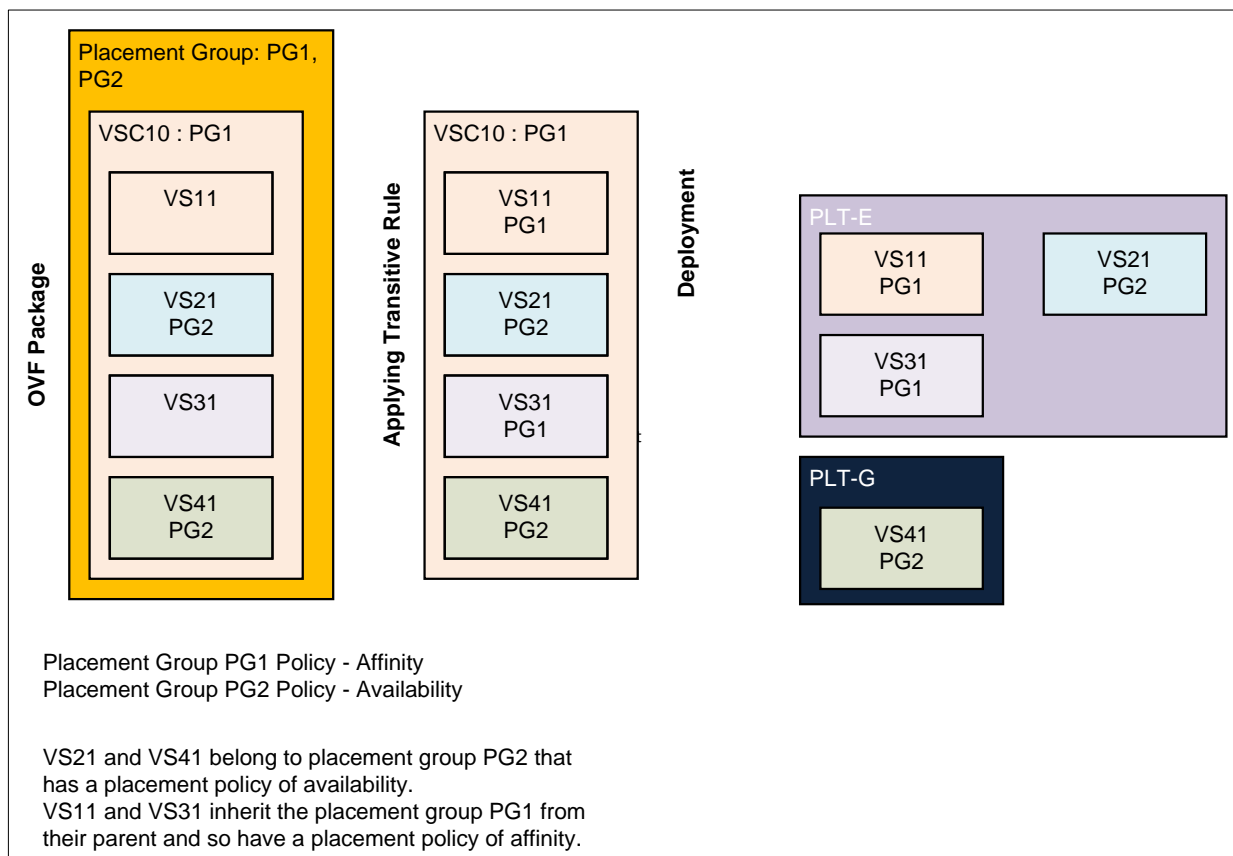
```

1160 <VirtualSystemCollection ovf:id="VSC10">
1161   <ovf:PlacementSection ovf:group="PG1">
1162     <Info>Placement policy group reference</Info>
1163   </ovf:PlacementSection>
1164
1165   <VirtualSystem ovf:id="VS11">
1166     <Info>Web server</Info>
1167     ...
1168   </VirtualSystem>
1169   <VirtualSystem ovf:id="VS21">
1170     <Info>Web server</Info>
1171     ...
1172     <ovf:PlacementSection ovf:group="PG2">
1173       <Info>Placement policy group reference</Info>
1174     </ovf:PlacementSection>
1175     ...
1176   </VirtualSystem>
1177 </VirtualSystemCollection>
1178
1179 <VirtualSystem ovf:id="VS31">
1180   <Info>Web server</Info>
1181   ...
1182 </VirtualSystem>
1183
1184 <VirtualSystem ovf:id="VS41">
1185   <Info>Web server</Info>
1186   ...
1187   <ovf:PlacementSection ovf:group="PG2">
1188     <Info>Placement policy group reference</Info>
1189   </ovf:PlacementSection>
1190   ...
1191 </VirtualSystem>
1192 </VirtualSystemCollection>

```

1190 In this example, the virtual systems, when instantiated, should be placed according to the placement
 1191 policies specified in the “PG1” and “PG2” placement groups. As shown in 4.4.8, the placement group ‘PG1’
 1192 has an affinity placement policy. As shown in 4.4.8, the placement group ‘PG2’ has an availability
 1193 placement policy. Figure 9 illustrates the placement of virtual systems in this more complex example. It
 1194 also illustrates the application of the transitivity rule.

1195



1196

1197

Figure 9 – Affinity and availability placement

1198 **4.5.6 EncryptionSection element**

1199 There are several reasons why it is desirable to have an encryption scheme enabling exchange of OVF
 1200 appliances while ensuring that only the intended consumers can use them. The encryption scheme
 1201 proposed in this specification utilizes existing encryption standards to incorporate this functionality in the
 1202 specification,

1203 The `EncryptionSection` element provides a single location for placing the encryption algorithm-related
 1204 markup and the corresponding reference list to point to the OVF content that has been encrypted.

1205 A document typically uses a single method of encryption, with a single key. However, the specification
 1206 allows the flexibility to encrypt different portions of the OVF descriptor with different keys derived by using
 1207 different methods and communicated to the end user in different ways.

1208 It is important to keep in mind that depending on which parts of the OVF descriptor have been encrypted,
 1209 an OVF descriptor may not validate against the OVF Schemas until decrypted.

1210 The encryption uses XML Encryption standard 1.1 to encrypt either the files in the reference section or
 1211 any parts of the XML markup of an OVF document.

1212 From an encryption standpoint, the important aspects that the standard defines are the

- 1213 a) algorithm used for the derivation of the key used in the encryption
- 1214 b) block encryption algorithm used to encrypt the content that uses the key
- 1215 c) method of transporting keys embedded in the OVF XML document.

1216 For each method of encryption used within the document, all the aspects that are necessary need to be
 1217 defined by the OVF package author. For instance, the author may choose to embed the key used in the
 1218 document, or the author may choose to communicate the key to desired end user by other means.

1219 The other aspect is a list of references to the markup sections in the OVF envelope, or the files in the
 1220 reference section that are encrypted by using the specific method. In order to be able to encrypt arbitrary
 1221 sections within the OVF Descriptor, use is made of the XML ID attribute within the ReferenceList
 1222 elements.

1223 The following example illustrates the conceptual structure of an Encryption section.

```

1224 <! --- Start of encryption section ---!>
1225   <! ---- Start of Markup for encryption method 1 ----!>
1226     <! ---- Markup defining key derivation aspects per XML encryption 1.1 ----!>
1227     <! ---- Markup defining the usage of the key for encryption per XML encryption 1.1
1228 ---!>
1229     <! ---- Optionally, the markup for key transportation per XML encryption 1.1 ---!>
1230     <! ---- Start of markup for pointers to the list of XML fragments encrypted using
1231 method 1----!>
1232       <! --- Pointer 1 ---!>
1233         .
1234         .
1235       <! --- Pointer N ---!>
1236     <! ---- End of markup for pointers to the list of XML fragments encrypted using
1237 method 1 ---!>
1238   <! ---- End of Markup for method 1 of encryption ----!>
1239
1240   <! ---- Start of the markup for encryption method N ----!>
1241     <! ---- Markup defining key derivation aspects per XML encryption 1.1 ----!>
1242     <! ---- Markup defining the usage of the key for encryption per XML encryption
1243 1.1 ---!>
1244     <! ---- Optionally, the markup for key transportation per XML encryption 1.1 -
1245 --!>
1246     <! ---- Start of markup for pointers to the list of XML fragments encrypted
1247 using method 1----!>
1248       <! --- Pointer 1 ---!>
1249         .
1250         .
1251       <! --- Pointer N ---!>
1252     <! ---- End of Markup for encryption method N ----!>
1253 <! --- End of encryption section ---!>

```

1254 Below is an example of an OVF encryption section with encryption methods utilized in the OVF
 1255 document, and the corresponding reference list pointing to the items that have been encrypted.

```

1256 <ovf:EncryptionSection>
1257 <!-- This section contains two different methods of encryption and the corresponding
1258 backpointers to the data that is encrypted. ->
1259 <!-- Method#1: Pass phrase based key derivation ->

```

```

1260 <!-- The following derived key block defines PBKDF2 and the corresponding
1261 backpointers to the encrypted data elements. -->
1262     <!-- Use a salt value "ovfpassword" and iteration count of 4096. --->
1263     <xenc11:DerivedKey>
1264         <xenc11:KeyDerivationMethod
1265 Algorithm="http://www.rsasecurity.com/rsalabs/pkcs/schemas/pkcs-5#pbkdf2"/>
1266 <pkcs-5:PBKDF2-params>
1267     <Salt>
1268         <Specified>ovfpassword</Specified>
1269     </Salt>
1270     <IterationCount>4096</IterationCount>
1271     <KeyLength>16</KeyLength>
1272     <PRF Algorithm="http://www.w3.org/2001/04/xmldsig-more#hmac-sha256"/>
1273 </pkcs-5:PBKDF2-params>
1274 ...
1275 <!-- The ReferenceList element below contains references to the file Ref-109.vhd via
1276 the URI syntax that is specified by XML Encryption.
1277 --->
1278 <xenc:ReferenceList>
1279     <xenc:DataReference URI="#first.vhd" />
1280 <xenc:DataReference URI=... />
1281 <xenc:DataReference URI=... />
1282 </xenc:ReferenceList>
1283 </xenc11:DerivedKey>
1284     <!-- Method#2: The following example illustrates use of a symmetric key
1285 transported by using the public key within a certificate. ->
1286 <xenc:EncryptedKey>
1287     <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-
1288 1_5"/>
1289     <ds:KeyInfo xmlns:ds='http://www.w3.org/2000/09/xmldsig#'
1290     <ds:X509Data>
1291     <ds:X509Certificate> ... </ds:X509Certificate>
1292     </ds:X509Data>
1293     </ds:KeyInfo>
1294     <xenc:CipherData>
1295     <xenc:CipherValue> ... </xenc:CipherValue>
1296     </xenc:CipherData>
1297 <!-- The ReferenceList element below contains reference #second-xml-fragment" to the
1298 XML fragment that has been encrypted by using the above method. --->
1299     <xenc:ReferenceList>
1300     <xenc:DataReference URI='#second-xml-fragment' />
1301     <xenc:DataReference URI='...' />
1302     <xenc:DataReference URI='...' />
1303     </xenc:ReferenceList>
1304     </xenc:EncryptedKey>
1305 </ovf:EncryptionSection>
1306 Below is an example of the encrypted file that is referenced in the EncryptionSection
1307 above by using URI='Ref-109.vhd' syntax.
1308 EXAMPLE:
1309 <ovf:References>
1310 <ovf:File ovf:id="Xen:9cb10691-4012-4aeb-970c-3d47a906bfff/0b13bdba-3761-8622-22fc-

```

```

1311 2e252ed9ce14" ovf:href="Ref-109.vhd">
1312 <!-- The encrypted file referenced by the package is enclosed by an EncryptedData with
1313 a CipherReference to the actual encrypted file. The EncryptionSection in this example
1314 has a backpointer to it under the PBKDF2 algorithm via Id="first.vhd". This tells the
1315 decrypter how to decrypt the file. -->
1316 <xenc:EncryptedData Id="first.vhd" Type='http://www.w3.org/2001/04/xmlenc#Element' >
1317     <xenc:EncryptionMethod
1318 Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc" />
1319     <xenc:CipherData>
1320         <xenc:CipherReference URI='Ref-109.vhd' />
1321     </xenc:CipherData>
1322 </xenc:EncryptedData>
1323 </ovf:File>
1324 </ovf:References>
1325 Below is an example of the encrypted OVF markup that is referenced in the
1326 EncryptionSection above by using URI='#second-xml-fragment' syntax.
1327 EXAMPLE:
1328 <!-- The EncryptedData element below encompasses encrypted xml from the original
1329 document. It is provided with the Id "second-xml-fragment", which allows it to be
1330 referenced from the EncryptionSection. -->
1331 <xenc:EncryptedData Type=http://www.w3.org/2001/04/xmlenc#Element Id="second-xml-
1332 fragment">
1333 <!-- Each EncryptedData specifies its own encryption method. -->
1334     <xenc:EncryptionMethod Algorithm=http://www.w3.org/2001/04/xmlenc#aes128-cbc/>
1335     <xenc:CipherData>
1336         <!-- Encrypted content --->
1337         <xenc:CipherValue>DEADBEEF</xenc:CipherValue>
1338     </xenc:CipherData>
1339 </xenc:EncryptedData>

```

1340

1341 4.6 OVF section elements used in virtual system collection

1342 4.6.1 ResourceAllocationSection element

1343 The ResourceAllocationSection element sets resource constraints that apply to a virtual system
 1344 collection. In contrast, the VirtualHardwareSection element applies to a specific virtual system.

1345 The ResourceAllocationSection element can use the bound attribute to set a minimum, a
 1346 maximum, or both for the allocation of a resource that applies in aggregate to all the virtual systems in the
 1347 virtual system collection.

1348 The following example illustrates a ResourceAllocationSection element. The processor allocation
 1349 illustrates the use of the bound attribute.

```

1350 <ResourceAllocationSection>
1351     <Info>Defines reservations for CPU and memory for the collection of VMs</Info>
1352     <Item>
1353         <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1354         <rasd:ElementName>300 MB reservation</rasd:ElementName>
1355         <rasd:InstanceID>0</rasd:InstanceID>
1356         <rasd:Reservation>300</rasd:Reservation>
1357         <rasd:ResourceType>4</rasd:ResourceType>

```

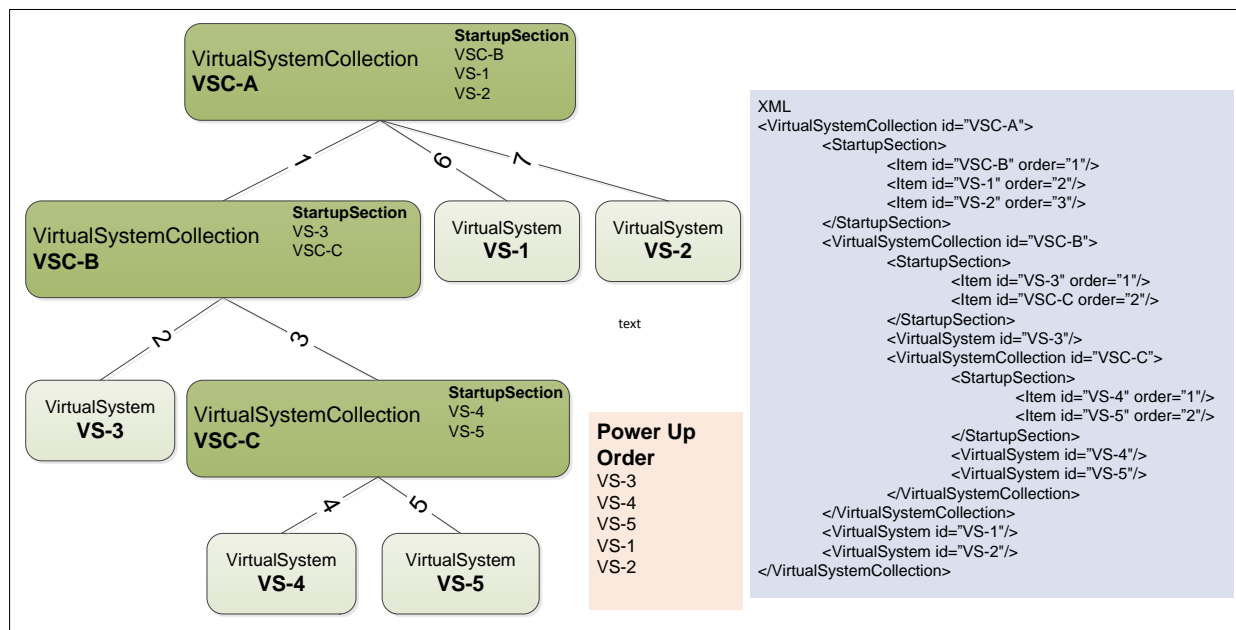
```

1358 </Item>
1359 <Item ovf:bound="min">
1360 <rasd:AllocationUnits>MHz</rasd:AllocationUnits>
1361 <rasd:ElementName>500 MHz reservation</rasd:ElementName>
1362 <rasd:InstanceID>1</rasd:InstanceID>
1363 <rasd:Reservation>500</rasd:Reservation>
1364 <rasd:ResourceType>3</rasd:ResourceType>
1365 </Item>
1366 <Item ovf:bound="max">
1367 <rasd:AllocationUnits>MHz</rasd:AllocationUnits>
1368 <rasd:ElementName>1500 MHz reservation</rasd:ElementName>
1369 <rasd:InstanceID>1</rasd:InstanceID>
1370 <rasd:Reservation>1500</rasd:Reservation>
1371 <rasd:ResourceType>3</rasd:ResourceType>
1372 </Item>
1373 </ResourceAllocationSection>
    
```

1374 **4.6.2 StartupSection element**

1375 The StartupSection element controls powering on and off of virtual system collections and is
 1376 executed after InstallSection element(s). The StartupSection element is a list of Item
 1377 elements. Item elements have attributes that control the order and timing of power on and power off. The
 1378 Item elements in a StartupSection element are scoped to that element. Do not confuse these
 1379 elements with Item elements in a VirtualHardwareSection element or
 1380 ResourceAllocationSection element.

1381 The Item elements within a StartupSection element reference either a VirtualSystem element or
 1382 a VirtualSystemCollection element. A StartupSection element may control powering up and
 1383 down of both virtual systems and virtual system collections contained in the virtual system collection
 1384 parent. This format allows for recursive startup structures. See Figure 10.



1385
1386 **Figure 10 – StartupSection traversal**

1387 The order of startup is determined by the value of the order attributes of the Item elements in a
 1388 StartupSection element in a VirtualSystemCollection element. The order attribute is a
 1389 nonnegative integer.

1390 If the `order` attribute of an `Item` element has a value of `'0'` (zero), the virtual system or virtual system
 1391 collection may be powered up at any time. The virtualization platform does not have to wait to start items
 1392 that have a higher order value.

1393 If the `order` attribute of an `Item` element has a nonzero value, the virtual systems are started in
 1394 ascending numeric order. Virtual systems that have the same value of the `order` attribute may be started
 1395 concurrently.

1396 It is recommended that virtual systems are stopped in descending numeric order. Virtual systems with
 1397 `Item` elements with an `order` attribute value of `'0'` (zero) may be stopped at any time.

1398 However, virtual systems are permitted to stop in a nondescending order in an implementation specific
 1399 manner unless the `shutdownorder` attribute is specified. The `shutdownorder` attribute allows the
 1400 shutdown order to be specified.

1401 Several optional attributes of the `Item` element support more detailed control of starting and stopping.
 1402 The `startDelay` and `stopDelay` attributes specify the seconds to wait until executing the next step in
 1403 the sequence. The default for both `startDelay` and `stopDelay` is zero.

1404 The `startAction` and `stopAction` attributes specify the actions to use in starting and stopping. Valid
 1405 values for `startAction` are `powerOn` and `none`. The default value is `'powerOn'`. Valid values for
 1406 the `stopAction` attribute are `'powerOn'`, `'guestShutdown'`, and `'none'`. The default value is
 1407 `'powerOff'`. If the `stopAction` attribute is set to `'guestShutdown'`, the action taken is deployment
 1408 platform specific.

1409 The `waitingForGuest` attribute is a Boolean that allows the deployment platform to wait until the guest
 1410 software reports readiness. The default value is `'FALSE'`. The communication mechanism is platform
 1411 specific.

1412 The following example illustrates a `StartupSection` element.

```
1413 <StartupSection>
1414   <Item ovf:id="vm1" ovf:order="0" ovf:startDelay="30" ovf:stopDelay="0"
1415     ovf:startAction="powerOn" ovf:waitingForGuest="true" ovf:stopAction="powerOff"/>
1416   <Item ovf:id="teamA" ovf:order="0"/>
1417   <Item ovf:id="vm2" ovf:order="1" ovf:startDelay="0" ovf:stopDelay="20"
1418     ovf:startAction="powerOn" ovf:stopAction="guestShutdown"/>
1419 </StartupSection>
```

1420 4.6.3 ScaleOutSection element

1421 The `ScaleOutSection` element allows dynamic configuration of the number of instantiated virtual
 1422 systems in a `VirtualSystemCollection` element. Without a `ScaleOut` element in the
 1423 `VirtualSystemCollection` element, the number of virtual systems and virtual system collections is
 1424 fixed. The `ScaleOutSection` element specifies a minimum and maximum number of replicas to be
 1425 created. At deployment time, the deployment platform chooses a value between the minimum and
 1426 maximum `InstanceCount`. The consumer can be queried for the value or the deployment platform can
 1427 make a determination based on other metadata. The `ScaleOutSection` element only appears in
 1428 `VirtualSystemCollection` elements, although both virtual system and virtual system collections may
 1429 be replicated.

1430 The following example illustrates a `ScaleOutSection` element.

```

1431 <VirtualSystemCollection ovf:id="web-tier">
1432   ...
1433   <ovf:ScaleOutSection ovf:id="web-server">
1434     <Info>Web tier</Info>
1435     <ovf:Description>Number of web server instances in web tier</ovf:Description>
1436     <ovf:InstanceCount ovf:default="4" ovf:minimum="2" ovf:maximum="8"/>
1437   </ovf:ScaleOutSection>
1438   ...
1439   <VirtualSystem ovf:id="web-server">
1440     <Info>Prototype web server</Info>
1441     ...
1442   </VirtualSystem>
1443 </VirtualSystemCollection>

```

1444 In the example above, the deployment platform creates a web tier that contains between two and eight
 1445 web server virtual machine instances, with a default count of four. The deployment platform makes an
 1446 appropriate choice (e.g., by prompting the user). Assuming three replicas were created, the OVF
 1447 environment that is available to the guest software in the first replica has the following content structure:

```

1448 <Environment ... ovfenv:id="web-server-1">
1449   ...
1450   <Entity ovfenv:id="web-server-2">
1451     ...
1452   </Entity>
1453   <Entity ovfenv:id="web-server-3">
1454     ...
1455   </Entity>
1456 </Environment>

```

1457 Note that the OVF `id` of the replicas is derived from the `id` of the prototype virtual system by adding a
 1458 sequence number. After deployment, all replica virtual systems have a sequence number suffix and no
 1459 virtual system has the base `id` of the prototype. If there is a `StartupSection` element, each replica has
 1460 the same startup number. It is not possible to specify a startup order among replicas.

1461 EXAMPLE:

```

1462 <VirtualSystemCollection ovf:id="web-tier">
1463   ...
1464   <DeploymentOptionSection>
1465     <Info>Deployment size options</Info>
1466     <Configuration ovf:id="minimal">
1467       <Label>Minimal</Label>
1468       <Description>Minimal deployment scenario</Description>
1469     </Configuration>
1470     <Configuration ovf:id="common" ovf:default="true">
1471       <Label>Typical</Label>
1472       <Description>Common deployment scenario</Description>
1473     </Configuration>
1474     ...
1475   </DeploymentOptionSection>
1476   ...
1477   <ovf:ScaleOutSection ovf:id="web-server">
1478     <Info>Web tier</Info>
1479     <ovf:Description>Number of web server instances in web tier</ovf:Description>
1480     <ovf:InstanceCount ovf:default="4"/>
1481     <ovf:InstanceCount ovf:default="1" ovf:configuration="minimal"/>
1482   </ovf:ScaleOutSection>
1483   ...
1484 </VirtualSystemCollection>

```

1485 In the example above, a `DeploymentOptionSection` element is used to control values for the
 1486 `InstanceCount` element in a `ScaleOutSection` element. Values in a `ScaleOutSection` element
 1487 can also be controlled through OVF property elements. The OVF properties are prompted for one time
 1488 for each replica. If the author wants an OVF property to be shared among replicas, it can be placed within
 1489 the `VirtualSystemCollection` element.

1490 4.7 OVF section elements used in virtual system

1491 4.7.1 OperatingSystemSection element

1492 The `OperatingSystemSection` element specifies the guest operating system used in a virtual system.
 1493 The selection of operating systems comes from the `CIM_OperatingSystem.OSType` property. The
 1494 OVF version and OVF id correspond to the `Value` and `ValueMap` of that property.

1495 The `id` attribute is required and refers to an integer from the `ValueMap`. The `version` attribute is
 1496 optional and refers to the corresponding `Value` for the `ValueMap`. The `version` attribute is a symbolic
 1497 string and cannot be internationalized.

1498 Both the `Info` (derived from `Section`) and `Description` elements may be externalized for localization.
 1499 See 5.2.

1500 This example is of a section that specifies a Microsoft Windows Server 2008:

```

1501 <OperatingSystemSection ovf:id="76" ovf:version="Microsoft Windows Server 2008">
1502   <Info>Specifies the operating system installed</Info>
1503   <Description>Microsoft Windows Server 2008</Description>
1504 </OperatingSystemSection>

```

1505 **4.7.2 InstallSection element**

1506 The `InstallSection` element is optional and is used only in `VirtualSystem` elements. If present, it is
1507 processed before the `StartupSection` element.

1508 The `InstallSection` elements enable the OVF package author to specify that a virtual system needs
1509 to reboot before powering off in order to complete the installation. Typically, when the boot occurs, the
1510 guest software executes scripts or other software from the OVF environment to complete the installation.
1511 The absence of an `InstallSection` element implies that a boot is not necessary to complete the
1512 installation. For example, if the virtual system has no guest software or the guest software is installed in
1513 the system image, an `InstallSection` element is not needed.

1514 The virtual systems in a virtual system collection may each have an `InstallSection` element defined.
1515 The reboots may be concurrent.

1516 The value of the `initialBootStopDelay` attribute is the duration in seconds that the virtualization
1517 platform waits for the virtual system to power off. If the delay expires and the virtual system has not
1518 powered off, the installation is deemed to have failed. The default value for `initialBootStopDelay` is
1519 zero, meaning that there is no limit on the delay and the virtualization platform waits until the virtual
1520 system powers itself off. The guest software on the virtual system could boot multiple times before
1521 powering off.

1522 In the example below, the virtualization platform waits 5 minutes (300 seconds) for the guest software to
1523 power off the virtual system. If the virtual machine does not power off in 5 minutes, the installation is
1524 deemed a failure. During the 5-minute wait interval, the virtual system could reboot several times.

```
1525 <InstallSection ovf:initialBootStopDelay="300">  
1526   <Info>Specifies that the virtual machine needs to be booted after having  
1527   created the guest software in order to install and/or configure the software  
1528   </Info>  
1529 </InstallSection>
```

1530 **4.7.3 EnvironmentFilesSection element**

1531 The `EnvironmentFilesSection` element allows the conveyance of additional environment files to the
1532 guest software permitting additional customization. These files are conveyed by using the same transport
1533 media as the OVF environment file.

1534 The OVF environment file is generated by the deployment function; however, any additional environment
1535 files are not and must be provided by the OVF package author. The additional environment files are
1536 specified in the `EnvironmentFilesSection` element with a `File` element that has an `ovf:fileRef`
1537 attribute and `ovf:path` attribute for each file.

1538 The `ovf:fileRef` attribute points to a `File` element in the `References` element. The `File` element is
1539 identified by matching its `ovf:id` attribute value with the `ovf:fileRef` attribute value.

1540 The `ovf:path` attribute indicates the relative location in the transport media where the file is placed.

```
1541 <Envelope>  
1542   <References>  
1543     ...  
1544     <File ovf:id="config" ovf:href="config.xml" ovf:size="4332"/>  
1545     <File ovf:id="resources" ovf:href="http://mywebsite/resources/resources.zip"/>  
1546   </References>  
1547   ...  
1548   <VirtualSystem ovf:id="...">
```

```

1549     ...
1550     <ovf:EnvironmentFilesSection ovf:required="false" ovf:transport="iso">
1551         <Info>Config files to be included in OVF environment</Info>
1552         <ovf:File ovf:fileRef="config" ovf:path="setup/cfg.xml"/>
1553         <ovf:File ovf:fileRef="resources" ovf:path="setup/resources.zip"/>
1554     </ovf:EnvironmentFilesSection>
1555     ...
1556 </VirtualSystem>
1557     ...
1558 </Envelope>

```

1559 In the example above, the file `config.xml` in the OVF package is copied to the OVF environment ISO
 1560 image and is accessible to the guest software in location `/ovffiles/setup/cfg.xml`, while the file
 1561 `resources.zip` is accessible in location `/ovffiles/setup/resources.zip` at deployment.

1562 4.7.4 BootDeviceSection element

1563 Earlier versions of OVF allowed virtual systems to boot only from the default boot device. This was found
 1564 to be a limitation in various scenarios that are encountered in OVF deployment.

- 1565 a) There was no way to specify whether a virtual system needed to be set up to PXE boot from a
 1566 NIC. Also, there was no way to specify whether a virtual system needed to be set up to boot
 1567 from a secondary disk or a USB device. Thus, there was a need to be able to specify these
 1568 alternative boot sources with their corresponding settings.
- 1569 b) A further need was identified, through implementation experience, to be able to specify multiple
 1570 alternative boot configurations. For instance during the “preparation” phase of the OVF, it may
 1571 be necessary for a virtual system to be patched by using a fix-up disk.

1572 The Common Information Model (CIM) defines artifacts to deal with boot order use cases prevalent in the
 1573 industry for BIOSes found in desktops and servers. The `CIM_BootSourceSetting` class defines an
 1574 individual boot source device, like an NIC or a disk, that is to be used as the boot source. Each of the
 1575 devices is identified by a unique ID specified in the `CIM_BootSourceSetting` class.

1576 A boot configuration is defined by a sequence of boot devices under an aggregation class,
 1577 `CIM_BootConfigSetting`. Thus a sequence of one or more `CIM_BootSourceSetting` properties is
 1578 aggregated into the `CIM_BootConfigSetting` class.

1579 The OVF envelope allows multiple such boot configurations to be aggregated into the
 1580 `BootDeviceSection` element. Each such `BootDeviceSection` element can be part of a
 1581 `VirtualHardwareSection` element.

1582 A deployment function attempts to set up a boot source sequence for a virtual system as defined in the
 1583 boot configuration that it has chosen. Choosing a boot configuration is an issue if there are more than one
 1584 boot configurations. The deployment function makes that choice based on the state of the deployment
 1585 and the caption element in the boot configuration structure.

1586 In the example below, the Pre-Install configuration specifies the boot source as a specific device
 1587 (network), while the Post-Install configuration specifies a device type (hard disk).

```

1588 EXAMPLE:
1589 <Envelope>
1590     ...
1591 <VirtualSystem ovf:id="...">
1592     ...
1593 <ovf:BootDeviceSection>
1594     <Info>Boot device order specification</Info>

```

```
1595     <bootc:CIM_BootConfigSetting>
1596     <bootc:Caption>Pre-Install</bootc:Caption>
1597     <bootc:Description>Boot Sequence for fixup of disk</bootc:Description>
1598     <boots:CIM_BootSourceSetting>
1599         <boots:Caption>Fix-up DVD on the network</boots:Caption>
1600         <boots:InstanceID>3</boots:InstanceID>           <!-- Network device-->
1601     </boots:CIM_BootSourceSetting>
1602     <boots:CIM_BootSourceSetting>
1603         <boots:Caption>Boot virtual disk</boots:Caption>
1604         <boots:StructuredBootString>CIM:Hard-Disk</boots:StructuredBootString>
1605     </boots:CIM_BootSourceSetting>
1606 </bootc:CIM_BootConfigSetting>
1607 </ovf:BootDeviceSection>
1608     ...
1609 </VirtualSystem>
1610 </Envelope>
```

1611 5 Authoring an OVF package

1612 5.1 Creation

1613 The creation of an OVF package involves

- 1614 i) packaging a set of VMs onto a set of virtual disks
- 1615 ii) encoding of those virtual disks appropriately
- 1616 iii) attaching an OVF descriptor with a specification of the virtual hardware, licensing, and other
1617 customization metadata
- 1618 iv) possibly including the attachment of a digital signature for the package

1619 The process of deploying an OVF package occurs when a virtualization platform consumes the OVF and
1620 creates a set of virtual machines from its contents.

1621 Creating an OVF can be made as simple as exporting an existing virtual machine from a virtualization
1622 platform into an OVF package, and adding to it the relevant metadata needed for correct installation and
1623 execution. This transforms the virtual machine from its current run-time state on a particular hypervisor
1624 into an OVF package. During this process, the virtual machine's disks can be compressed to make them
1625 more convenient to distribute.

1626 For commercial-grade virtual appliances, a standard build environment can be used to produce an OVF
1627 package. For example, the OVF descriptor can be managed by using a source control system, and the
1628 OVF package can be built by using a reproducible scripting environment (such as `make` files) or through
1629 the use of appliance building toolkits that are available from multiple vendors.

1630 When an OVF package is created, it can be accompanied with appliance-specific, post-installation
1631 configuration metadata. This includes metadata for optional localization of the interface language(s) of the
1632 appliance, review/signoff and/or enforcement of the EULA, and resource configuration. It can also involve
1633 the addition of special drivers, agents, and other tools to the guest to enhance (for example) I/O,
1634 timekeeping, memory management, monitoring, and ordered shutdown.

1635 The process of authoring the OVF descriptor is essentially putting together the building blocks for the
1636 virtual appliance. As indicated earlier, a virtual appliance is defined by the description of the virtual
1637 systems composing the appliance, metadata regarding the appliance and the guest software, and a set of
1638 referenced files. The OVF descriptor is the central element to aggregate and reference all required

1639 information. The major building blocks of the OVF descriptor are sections. Clause 4 introduces the
 1640 various sections that can be used to describe the virtual appliance.

1641 5.2 Internationalization

1642 The OVF specification supports localizable messages by using the optional `ovf:msgid` attribute.
 1643 Localized messages can be used to display the user messages in the local language during deployment.

```
1644 <Envelope ...>
1645   ...
1646   <Info ovf:msgid="info.os">Operating System</Info>
1647   ...
1648   <Strings xml:lang="da-DA">
1649     <Msg ovf:msgid="info.os">Operativsystem</Msg>
1650     ...
1651   </Strings>
1652   <Strings xml:lang="de-DE">
1653     <Msg ovf:msgid="info.os">Betriebssystem</Msg>
1654     ...
1655   </Strings>
1656 </Envelope>
```

1657 The example above defines an `Info` element within a section. The information in this section is related to
 1658 the operating system of the virtual system. The attribute `ovf:msgid="info.os"` indicates that the
 1659 string between the start-tag and the end-tag of the `Info` element can be replaced with a localized
 1660 message. The localized message is referred to by its message ID of `info.os`. If there is a suitable
 1661 localized message set in a `Strings` section, the default message “Operating System” is replaced by the
 1662 localized message taken from the `Strings` section corresponding to the current region.

1663 In the example above the localized strings are stored inside the OVF descriptor. Localized strings can
 1664 also be stored outside the OVF descriptor by using external string bundles. For example:

```
1665 <Envelope ...>
1666   <References>
1667     ...
1668     <File ovf:id="da-DA-resources" ovf:href="danish.msg"/>
1669     <File ovf:id="de-DE-resources" ovf:href="german.msg"/>
1670     ...
1671   </References>
1672   ...
1673   <Info ovf:msgid="info.os">Operating System</Info>
1674   ...
1675   <Strings xml:lang="da-DA" ovf:fileRef="da-da-resources"/>
1676   <Strings xml:lang="de-DE" ovf:fileRef="de-de-resources"/>
1677 </Envelope>
```

1678 The localized message for “Operating System” is defined in the files `danish.msg` and `german.msg`. The
 1679 format of the external message file `german.msg` is described in the example below.

```
1680 <Strings
1681   xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1"
1682   xmlns="http://schemas.dmtf.org/ovf/envelope/1"
1683   xml:lang="de-DE">
1684   ...
```

```

1685 <Msg ovf:msgid="info.os">Betriebssystem</Msg>
1686 ...
1687 </Strings>

```

1688 In the top `Strings` section, the `xml:lang` attribute is used to define the locale of the particular external
 1689 message file. The external message file contains `Msg` elements for the localized messages that are used
 1690 in the OVF descriptor.

1691 Another method of using localized resources is to reference external files based on the current location.
 1692 This method can be used, for example, to display a license text based on the location. The license text is
 1693 contained in a text file per location. The following example shows how to reference an external plain text
 1694 file to display a localized license.

```

1695 <Envelope xml:lang="en-US">
1696   <References>
1697     <File ovf:id="license-en-US" ovf:href="license-en-US.txt"/>
1698     <File ovf:id="license-de-DE" ovf:href="license-de-DE.txt"/>
1699   </References>
1700   ...
1701   <VirtualSystem ovf:id="...">
1702     <EulaSection>
1703       <Info>Licensing agreement</Info>
1704       <License ovf:msgid="license">Unused</License>
1705     </EulaSection>
1706     ...
1707   </VirtualSystem>
1708   ...
1709   <Strings xml:lang="en-US">
1710     <Msg ovf:msgid="license" ovf:fileRef="license-en-US">Invalid license</Msg>
1711   </Strings>
1712   <Strings xml:lang="de-DE">
1713     <Msg ovf:msgid="license" ovf:fileRef="license-de-DE">Ihre Lizenz ist nicht
1714     gültig</Msg>
1715   </Strings>
1716 </Envelope>

```

1717 The `License` element contains an `ovf:msgid` attribute. In the `Strings` sections, the `ovf:msgid` for
 1718 the different locations is linked to a file reference by using the `ovf:fileRef` attribute. The
 1719 `ovf:fileRef` attribute has a corresponding entry in the `References` section of the OVF descriptor.
 1720 The entry in the `References` section resolves to an external text file that contains the license text.

1721 5.3 Extensibility

1722 The OVF specification allows custom metadata to be added to OVF descriptors in several ways:

- 1723 • New section elements may be defined as part of the `Section` substitution group, and used
 1724 wherever the OVF Schemas allow sections to be present.
- 1725 • The OVF Schemas use an open content model, where all existing types may be extended at the
 1726 end with additional elements. Extension points are declared in the OVF Schemas with `xs:any`
 1727 declarations with `namespace="##other"`.
- 1728 • The OVF Schemas allow additional attributes on existing types.

1729 A design goal of the OVF specification is to ensure backward and forward compatibility. For forward
 1730 compatibility, this means that an OVF descriptor using features of a later specification (or custom
 1731 extensions) can be understood by an OVF consumer that either i) is written to an earlier version of the
 1732 specification, or ii) has no knowledge of the particular extensions. The OVF consumer should be able to
 1733 reliably, predictably, and in a user-friendly manner, decide whether to reject or accept an OVF package
 1734 that contains extensions.

1735 5.3.1 Substitution group

1736 OVF supports an open-content model that allows additional sections to be added, as well as allowing
 1737 existing sections to be extended with new content. On extensions, a Boolean `ovf:required` attribute
 1738 specifies whether the information in the element is required for correct behavior or is optional.

1739 Additional sections can be inserted into the OVF descriptor by defining new members of the
 1740 `ovf:Section` substitution group. This means the new section extends the base schema for a `Section`
 1741 element. New sections can be used to define metadata that is not related to the existing sections defined
 1742 in the OVF specification. The new `Section` has an `<Info>` element that is used to display information to
 1743 the consumer regarding the section in case the deployment function does not understand the section.

1744 The example shows the addition of a new `Section` `<ns:BuildInformationSection>`. The
 1745 `Section` uses the namespace `ns`. The namespace is referenced in a parent element e.g., in the
 1746 `<Envelope>` element: `<Envelope xmlns="http://schemas.dmtf.org/ovf/envelope/2"`
 1747 `xmlns:ns="http://acme.org/ovf/extension/ns">`. As required by the `ovf:Section`
 1748 `substitutionGroup`, the new section contains an `<Info>` element. The elements `BuildNumber`,
 1749 `BuildDate`, `BuildSystem` are new elements. The elements are defined in the referenced namespace
 1750 schema. The `ovf:required` attribute is set to "false" to indicate that the deployment function warns but
 1751 does not fail if it cannot implement the section.

1752 Example of adding a new section:

```
1753 <ns:BuildInformationSection ovf:required="false">
1754   <Info>Specifies information about how a virtual machine was created</Info>
1755   <BuildNumber> ... </BuildNumber >
1756   <BuildDate> ... </BuildDate >
1757   <BuildSystem> ... </BuildSystem>
1758   ...
1759 </ns:BuildInformationSection>
```

1760 The XSD schema for the additional section in the example above looks as follows.

```
1761 <?xml version="1.0" encoding="UTF-8"?>
1762 <xs:schema xmlns:ns=http://acme.org/ovf/extension/ns
1763   xmlns:ovf=http://schemas.dmtf.org/ovf/envelope/2
1764   xmlns:xs=http://www.w3.org/2001/XMLSchema
1765   targetNamespace=http://acme.org/ovf/extension/ns
1766   elementFormDefault="qualified"
1767   attributeFormDefault="qualified">
1768 <xs:import namespace=http://schemas.dmtf.org/ovf/envelope/2
1769   schemaLocation="dsp8023.xsd"/>
1770   <xs:element name="BuildInformationSection" type="ns:BuildInformationSection_Type"
1771     substitutionGroup="ovf:Section">
1772     <xs:annotation>
1773       <xs:documentation>Element substitutable for Section since
1774       BuildInformationSection_Type is a derivation of Section_Type
1775     </xs:documentation>
```

```

1776     </xs:annotation>
1777 </xs:element>
1778 <xs:complexType name="BuildInformationSection_Type">
1779 <xs:annotation>
1780   <xs:documentation>An ACME specific section.</xs:documentation>
1781 </xs:annotation>
1782 <xs:complexContent>
1783   <xs:extension base="ovf:Section_Type">
1784     <xs:sequence>
1785       <xs:element name="BuildNumber" maxOccurs="unbounded">
1786         <xs:complexType>
1787           <xs:anyAttribute namespace="##any" processContents="lax"/>
1788         </xs:complexType>
1789       </xs:element>
1790       <xs:element name="BuildDate" maxOccurs="unbounded">
1791         <xs:complexType>
1792           <xs:anyAttribute namespace="##any" processContents="lax"/>
1793         </xs:complexType>
1794       </xs:element>
1795       <xs:element name="BuildSystem" maxOccurs="unbounded">
1796         <xs:complexType>
1797           <xs:anyAttribute namespace="##any" processContents="lax"/>
1798         </xs:complexType>
1799       </xs:element>
1800     </xs:sequence>
1801     <xs:anyAttribute namespace="##any" processContents="lax"/>
1802   </xs:extension>
1803 </xs:complexContent>
1804 </xs:complexType>
1805 </xs:schema>

```

1806 The schema defines a `BuildInformationSection` substitution group for the `ovf:Section` section.
 1807 The `BuildInformationSection` substitution group is of the `BuildInformationSection_Type`
 1808 type. `BuildInformationSection_Type` type defines `ovf:Section_Type` as a base type and
 1809 extends the `ovf:Section_Type` by the `BuildNumber`, `BuildDate` and `BuildSystem` elements.

1810 5.3.2 Elements

1811 New elements within existing sections can be added at the end of the section. The `Envelope`,
 1812 `VirtualSystem`, `VirtualSystemCollection`, `Content` and `Strings` sections do not support the
 1813 addition of elements at the end of the section. The used namespace needs to be referenced in a parent
 1814 element and must be different from the OVF namespace. Additional elements can be used to extend the
 1815 information given for a particular section in the OVF descriptor.

1816 An illustration of extending an existing section is given below.


```

1817 <AnnotationSection>
1818   <Info>Specifies an annotation for this virtual machine</Info>
1819   <Annotation>This is an example of how a future element (Author) can still be
1820   parsed by older clients</Annotation>
1821   <!-- AnnotationSection extended with Author element -->
1822   <ns:Author ovf:required="false">John Smith</ns:Author>
1823 </AnnotationSection>

```

1824 The example shows an additional element in the `Annotation` section. The element extends the
 1825 `Annotation` section with information regarding the `Author` of the descriptor. The new element belongs
 1826 to the `ns` namespace.

1827 5.3.3 Attributes

1828 A third option of extending an OVF descriptor with additional information is to add custom attributes into
 1829 existing elements. These attributes can be used to extend the information given by an existing element.

```

1830 <!-- Optional custom attribute example -->
1831   <Network ovf:name="VM network" ns:desiredCapacity="1 Gbit/s">
1832     <Description>The main network for VMs</Description>
1833   </Network>

```

1834 The example above shows the addition of a `desiredCapacity` attribute for the `Network` element. The
 1835 new attribute is defined in the `ns` namespace.

1836 See ANNEX E for more detailed examples of OVF document extensions.

1837 5.4 Conformance

1838 The OVF specification defines three conformance levels for OVF descriptors, with 1 being the highest
 1839 level of conformance:

- 1840 • OVF descriptor only contains metadata defined in the OVF specification, i.e., no custom
 1841 extensions are present.
 1842 Conformance Level: 1.
- 1843 • OVF descriptor contains metadata with custom extensions, but all such extensions are optional.
 1844 Conformance Level: 2.
- 1845 • OVF descriptor contains metadata with custom extensions, and at least one such extension is
 1846 required.
 1847 Conformance Level: 3.

1848 The use of conformance level 3 limits portability, which means that the OVF package might not be
 1849 deployed on any other virtualization platform other than the one supporting the custom extensions.

1850 5.5 Virtual hardware description

1851 The hardware description shown below is very general. In particular, it specifies that a virtual disk and a
 1852 network adaptor is needed. It does not specify what the specific hardware should be. For example, a
 1853 SCSI or IDE disk, or an E1000 or Vlance network card is appropriate. More specifically, it can be
 1854 assumed that if the specification is generic, the appliance undertakes discovery of the devices present,
 1855 and loads relevant drivers. In this case, it is assumed that the appliance creator has developed the
 1856 appliance with a broad set of drivers, and has tested the appliance on relevant virtual hardware to ensure
 1857 that it works.

1858 If an OVF package is deployed on a virtualization platform that does not offer the same hardware devices
1859 and/or categories of devices that are required by the guest software that is included in the appliance,
1860 installation failures that are nontrivial and nonobvious can occur. The risk is that it fails to install and/or
1861 boot, and that the user is not able to debug the problem. With this situation comes the risk of increased
1862 volume in customer support calls, and general customer dissatisfaction. A more constrained and detailed
1863 virtual hardware specification can reduce the chance of incorrect execution (because the specific devices
1864 required are listed), but this specification limits the number of systems that the appliance may be able
1865 install on and/or boot.

1866 Consider that simplicity, robustness, and predictability of installation are key reasons that ISVs are
1867 moving to the virtual appliance model. Therefore, appliance developers should create appliances for
1868 which the hardware specification is more rather than less generic, unless the appliance has very specific
1869 hardware needs. At the outset, the portability of the appliance is based on the guest software used in the
1870 virtual machines and the range of virtual hardware the guest software supports.

1871 Ideally, the appliance vendor creates a virtual machine that has device drivers for the virtual hardware of
1872 all of the vendor's desired target virtualization platforms. However, many virtualization platform vendors
1873 today do not distribute drivers independently to virtual appliance vendors/creators. Instead, to further
1874 simplify the management of the virtual hardware-to-appliance interface, the OVF model supports an
1875 explicit installation mode, in which each virtual machine is booted once right after installation, to permit
1876 localization/customization for the specific virtualization platform. This mode allows the virtual machine to
1877 detect the virtualization platform and install the correct set of device drivers, including any platform-
1878 specific drivers that are made available to the guest when it first reboots (for example, via floppy or CD
1879 drives attached to the guest on first boot). In addition, for sysprepped Windows VMs, that need only re-
1880 installation and customization with naming etc., the reboot technique allows naming and tailoring of the
1881 image to be achieved without consumer intervention.

1882 The following example illustrates multiple virtual hardware profiles for different virtualization platforms
1883 specified in the same descriptor.

```
1884 <VirtualHardwareSection>  
1885   <Info>500Mb, 1 CPU, 1 disk, 1 nic virtual machine, Platform A</Info>  
1886   <System>  
1887     ...  
1888   </System>  
1889   <Item>  
1890     ...  
1891   </Item>  
1892   ...  
1893 </VirtualHardwareSection>  
1894 <VirtualHardwareSection>  
1895   <Info>500Mb, 1 CPU, 1 disk, 1 nic virtual machine, Platform B</Info>  
1896   <System>  
1897     ...  
1898   </System>  
1899   <Item>  
1900     ...  
1901   </Item>  
1902   ...  
1903 </VirtualHardwareSection>
```

1904 This type of profile allows the vendor to tailor the hardware description to support different virtualization
1905 platforms and features. A specific virtualization platform may choose between any of the specific virtual
1906 hardware sections that it can support, with the assumption that the OVF deployment function chooses the
1907 latest or most capable feature set that is available on the local platform.

1908 The example below shows how a specific type of virtual hardware can be defined. Multiple options for the
 1909 `rasd:ResourceSubType` can be separated by a single space character. The deployment function can
 1910 then choose the virtual hardware type to instantiate.

1911 If multiple resource subtypes are specified in the OVF package, the deployment function selects an
 1912 appropriate value to populate the `CIM_ResourceAllocationSettingData` class for the managed
 1913 environment. If more than one appropriate value exists the deployment function selects any of the
 1914 appropriate values. If the deployment function does not find an appropriate value the deployment fails.

```

1915 <Item>
1916     <rasd:ElementName>SCSI Controller 0</rasd:ElementName>
1917     <rasd:InstanceID>1000</rasd:InstanceID>
1918     <rasd:ResourceSubType>LsiLogic BusLogic</rasd:ResourceSubType>
1919     <rasd:ResourceType>6</rasd:ResourceType>
1920 </Item>
1921 <Item>
1922     <rasd:ElementName>Harddisk 1</rasd:ElementName>
1923     <rasd:HostResource>ovf:/disk/vmdisk1</rasd:HostResource>
1924     <rasd:InstanceID>22001</rasd:InstanceID>
1925     <rasd:Parent>1000</rasd:Parent>
1926     <rasd:ResourceType>17</rasd:ResourceType>
1927 </Item>

```

1928 5.6 Example descriptors

1929 The following examples have been provided as complete examples of an OVF descriptor. These
 1930 examples pass XML validation.

1931 ANNEX A illustrates an OVF descriptor for a single virtual system.

1932 ANNEX B illustrates a multiple virtual system OVF descriptor.

1933 ANNEX C illustrates an OVF descriptor for a single virtual system with multiple applications contained in
 1934 it; i.e., a LAMP stack.

1935 ANNEX D illustrates an OVF descriptor for a multiple virtual system with multiple applications contained in
 1936 it, i.e., a LAMP stack with two virtual systems.

1937 6 Deploying an OVF package

1938 6.1 Deployment

1939 Deployment transforms the virtual machines in an OVF package into the run-time format understood by
 1940 the target virtualization platform, with the appropriate resource assignments and supported by the correct
 1941 virtual hardware. During deployment, the platform validates the OVF integrity, making sure that the OVF
 1942 package has not been modified in transit, and checks that it is compatible with the local virtual hardware.
 1943 It also assigns resources to, and configures the virtual machines for, the particular environment on the
 1944 target virtualization platform. This process includes assigning and configuring the networks (physical and
 1945 virtual) too which the virtual machines are connected; assigning storage resources for the VMs, including
 1946 virtual hard disks, as well as any transient data sets, connections to clustered or networked storage and
 1947 the like; configuring CPU and memory resources; and customizing application level properties. OVF does
 1948 not support the conversion of guest software between processor architectures or hardware platforms.
 1949 Deployment instantiates one or more virtual machines with a hardware profile that is compatible with the
 1950 requirements captured in the OVF descriptor, and a set of virtual disks with the content specified in the
 1951 OVF package.

1952 The deployment experience of an OVF package depends on the virtualization platform on which it is
1953 deployed. It could be command-line based, scripted, or a graphical deployment wizard. The typical OVF
1954 deployment tool shows, or prompts for, the following information:

- 1955 • Show information about the OVF package (from the `ProductSection`), and ask the user to
1956 accept the licensing agreement, or address an unattended installation.
- 1957 • Validate that the virtual hardware is compatible with the specification in the OVF.
- 1958 • Ask the user for the storage location of the virtual machines and the physical networks to which
1959 the logical networks in the OVF package are connected.
- 1960 • Ask the user to enter the specific values for the properties configured in the `ProductSection`.

1961 After this configuration, it is expected that the virtual machines can be started successfully to obtain
1962 (using standard procedures such as DHCP) an identity that is valid on the local network. Properties are
1963 used to prompt for specific IP network configuration and other values that are particular to the deployment
1964 environment. After the appliance is booted for the first time, additional configuration of software inside the
1965 appliance can be done through a management interface provided by the appliance itself, such as a web
1966 interface.

1967 **6.2 OVF environment descriptor**

1968 The OVF environment descriptor is an XML document that describes metadata about the software
1969 installed on the virtual disks. The OVF specification defines the common sections used for deploying
1970 software, such as virtual hardware, disks, networks, resource requirements, and customization
1971 parameters. The descriptor is designed to be extensible so further information can be added later.

1972 A virtual appliance often needs to be customized to function properly in the particular environment where
1973 it is deployed. The OVF environment provides a standard and extensible way for the virtualization
1974 platform to communicate deployment configuration to the guest software.

1975 The OVF environment is an XML document containing deployment time customization information for the
1976 guest software. Examples of information that could be provided in the XML document include:

- 1977 • Operating system level configuration, such as host names, IP address, subnets, gateways, etc.
- 1978 • Application-level configuration, such as DNS name of active directory server, databases, and
1979 other external services

1980 The set of properties that are to be configured during deployment is specified in the OVF descriptor by
1981 using the `ProductSection` metadata, and is entered by the user using a wizard style interface during
1982 deployment.

1983 For instance, the OVF environment allows guest software to automate the network settings between
1984 multitiered services, and the web server may automatically configure itself with the IP address of the
1985 database server without any manual user interaction.

1986 Defining a standard OVF environment does pose some challenges, because no standard cross-vendor
1987 para-virtualized device exists for communicating between the guest software in a virtual machine and the
1988 underlying virtualization platform. The approach taken by the OVF specification is to split the OVF
1989 environment definitions into two parts:

- 1990 • a standard *protocol* that specifies what information is available and in what format it is available
- 1991 • a *transport* that specifies how the information is obtained

1992 The specification requires all implementations to support an ISO transport that makes the OVF
1993 environment (XML document) available to the guest software on a dynamically generated ISO image.

1994 6.3 Resource configuration options during deployment

1995 The OVF package has the ability to include resource configuration options for a virtual appliance. This
 1996 makes it easy for the package consumer to get an initial setup without having to make individual resource
 1997 decisions based on the intended use. The `Description` and `Label` elements provide a human-
 1998 readable list of resource configurations, for example:

- 1999 • Software evaluation setup
- 2000 • 10-100 person workgroup setup
- 2001 • 100-1000 person workgroup setup
- 2002 • Large enterprise workgroup setup

2003 The deployment function prompts for selection of a configuration. The list of configurations described
 2004 above is expected to be used for a suitable initial resource configuration.

2005 Example list of configurations:

```
2006 <DeploymentOptionSection>
2007   <Configuration ovf:id="eval">
2008     <Label>Software Evaluation</Label>
2009     <Description>Software evaluation setup</Description>
2010   </Configuration>
2011   <Configuration ovf:id="small" ovf:default="yes">
2012     <Label>Small</Label>
2013     <Description>10-100 person workgroup setup</Description>
2014   </Configuration>
2015   <Configuration ovf:id="medium">
2016     <Label>Medium</Label>
2017     <Description>100-1000 person workgroup setup</Description>
2018   </Configuration>
2019   <Configuration ovf:id="large">
2020     <Label>Large</Label>
2021     <Description>Large enterprise workgroup setup</Description>
2022   </Configuration>
2023 </DeploymentOptionSection>
```

2024 The following snippet of an OVF descriptor illustrates the configuration option use for a resource
 2025 requirement. In this case, if the consumer chose 'eval', the resource allocation data used is that in the
 2026 Item `ovf:configuration="eval"`.

2027 Resource requirement example:

```
2028 <ResourceAllocationSection>
2029   <Info>Defines reservations for CPU and memory</Info>
2030   <Item>
2031     ... default configuration ...
2032   </Item>
2033   <Item ovf:configuration="eval">
2034     ... replaces the default configuration if the "eval" configuration if selected
2035   ...
2036   </Item>
2037 </ResourceAllocationSection>
```

2038 The following snippet of an OVF descriptor illustrates the configuration option use for a
2039 VirtualHardwareSection. In this case, if the consumer chose “large”, the resource allocation data
2040 used is that in the Item `ovf:configuration="large"`.

2041 VirtualHardwareSection example:

```
2042 <VirtualHardwareSection>
2043   <Info>...</Info>
2044   <Item>
2045     <rasd:AllocationUnits>hertz * 10^6</rasd:AllocationUnits>
2046     <rasd:ElementName>1 CPU and 500 MHz reservation</rasd:ElementName>
2047     <rasd:InstanceID>1</rasd:InstanceID>
2048     <rasd:Reservation>500</rasd:Reservation>
2049     <rasd:Limit>1100</rasd:Reservation>
2050     <rasd:ResourceType>3</rasd:ResourceType>
2051     <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2052   </Item>
2053   ...
2054   <Item ovf:configuration="large">
2055     <rasd:AllocationUnits>hertz * 10^6</rasd:AllocationUnits>
2056     <rasd:ElementName>1 CPU and 800 MHz reservation</rasd:ElementName>
2057     <rasd:InstanceID>1</rasd:InstanceID>
2058     <rasd:Reservation>800</rasd:Reservation>
2059     <rasd:ResourceType>3</rasd:ResourceType>
2060     <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2061   </Item>
2062 </VirtualHardwareSection>
```

2063 6.4 Product customization during deployment using Property elements

2064 The OVF descriptor can contain a description of the guest software that includes information about
2065 customization provided through the OVF environment. This information is provided by the use of
2066 Property elements in the ProductSection of the OVF descriptor.

2067 Each Property element has five possible attributes: `ovf:key`, `ovf:type`, `ovf:qualifiers`,
2068 `ovf:value`, and `ovf:userConfigurable`.

2069 The `ovf:key` attribute is a unique identifier for the Property element.

2070 The `ovf:type` attribute indicates the type of value the Property element contains.

2071 The `ovf:qualifiers` attribute specifies additional information regarding the `ovf:type` attribute so that
2072 CIM value maps can be used.

2073 The `ovf:value` attribute is used to provide a value for a Property element.

2074 The `ovf:userConfigurable` attribute determines whether the value provided is a default value and
2075 changeable at deployment or is not changeable.

2076 An example of the use of Property elements follows.

```

2077 <ProductSection>
2078   <Info>Describes product information for the service</Info>
2079   <Product>MyService Web Portal</Product>
2080   <Vendor>Some Random Organization</Vendor>
2081   <Version>4.5</Version>
2082   <FullVersion>4.5-b4523</FullVersion>
2083   <ProductUrl>http://www.vmware.com/go/ovf</ProductUrl>
2084   <VendorUrl>http://www.vmware.com/</VendorUrl>
2085   <Property ovf:key="adminEmail" ovf:type="string"
2086     ovf:userConfigurable="true">
2087     <Description>Email address of administrator</Description>
2088   </Property>
2089   <Property ovf:key="appIp" ovf:type="string"
2090     ovf:userConfigurable="true">
2091     <Description>IP address of the application</Description>
2092   </Property>
2093   <Property ovf:key="Gateway" ovf:type="string" ovf:value="192.168.0.1"
2094     ovf:userConfigurable="false" >
2095     <Description>Gateway address to be used</Description>
2096   </Property>
2097   <Property ovf:key=" SoftwareResourceType" ovf:type="uint16"
2098     ovf:qualifiers="uint16,uint16,uint16,uint16,uint16,uint16,uint16,uint16,uint16,
2099     uint16,"
2100     ovf:value="Unknown", "Buffer", "Queue", "Protocol Endpoint", "Remote Interface",
2101     "Pool", "Cache", "File", "Database",
2102     ovf:userConfigurable="false" >
2103     <Description>Value Map example based on SoftwareResourceType property in
2104     CIM_SoftwareResource class</Description>
2105   </Property>
2106 </ProductSection>

```

2107 In the CIM_SoftwareResource class, there is a property SoftwareResourceType that is shown in
 2108 the snippet from the CIM Schema below.

```

2109   [Description (
2110     "The type of the software resource. Although the behavior "
2111     "of the different software resource types is modeled "
2112     "similarly, different names for resources transferring "
2113     "data over time or/and space have been established. "
2114     "SoftwareResourceType conveys their original, most common "
2115     "name. \n"
2116     ValueMap { "0", "2", "3", "4", "5", "6", "7", "8", "9",
2117     "10..32767", "32768..65535" },
2118     Values { "Unknown", "Buffer", "Queue", "Protocol Endpoint",
2119     "Remote Interface", "Pool", "Cache", "File", "Database",
2120     "DMTF Reserved", "Vendor Reserved" }}
2121   uint16 SoftwareResourceType;

```

2122 The ovf:type corresponds to the index of the ValueMap so it is uint16.

2123 ANNEX D contains a detailed example of customization of a complex multitiered application.

2124 7 Portability

2125 OVF is an enabling technology for enhancing portability of virtual appliances and their associated virtual
2126 machines. An OVF package contains a recipe for creating virtual machines that can be interpreted
2127 concisely by a virtualization platform. The packaged metadata enables a robust and user-friendly
2128 experience when installing a virtual appliance. In particular, the metadata can be used by the
2129 management infrastructure to confidently decide whether a particular VM described in an OVF can be
2130 installed or whether it is rejected, and potentially to guide appropriate conversions and localizations to
2131 make it useable in the specific execution context that it is to be installed.

2132 There are many factors that are beyond the control of the OVF format specification, and even a fully
2133 compliant implementation of it, that determine the portability of a packaged virtual machine. That is, the
2134 act of packaging a virtual machine into an OVF package does not guarantee universal portability or
2135 installability across all hypervisors. Below are some of the factors that could limit portability:

- 2136 • The VMs in the OVF could contain virtual disks in a format that is not understood by the
2137 hypervisor attempting the installation. While it is reasonable to expect that most hypervisors are
2138 able to import and export VMs in any of the major virtual hard disk formats, newer formats may
2139 arise that are supported by the OVF and not a particular hypervisor.
- 2140 • The installed guest software may not support the virtual hardware presented by the hypervisor.
2141 By way of example, the Xen hypervisor does not by default offer a virtualized floppy disk device
2142 to guests. One could conceive of a guest VM that requires interaction with a floppy disk
2143 controller and therefore it is not able to execute the VM correctly.
- 2144 • The installed guest software does not support the CPU architecture. For example, the guest
2145 software might execute CPU operations specific to certain processor models or require specific
2146 floating point support, or contain opcodes specific to a particular vendor's CPU.
- 2147 • The virtualization platform might not understand a feature requested in the OVF descriptor. For
2148 example, composed services may not be supported. Because the OVF standard evolves
2149 independently of virtualization products, at any point an OVF might be unsupported on a
2150 virtualization platform that predates that OVF specification.

2151 The portability of an OVF can be categorized into the following classes:

- 2152 • **Portability class 1.** Runs on multiple families of virtual hardware. For example, the appliance
2153 could be used on Xen, Sun, Microsoft, and VMware hypervisors. For level 3 compatibility, the
2154 guest software has been developed to support the devices of multiple hypervisors. A clean
2155 install and boot of guest software, during which the guest software performs hardware device
2156 discovery and installs any specialized drivers required to interact with the virtual platform, is an
2157 example of Level 3 portability of an OVF. The “sysprep” level of portability for Microsoft
2158 Windows® operating systems is another example. Such guest software instances can be re-
2159 installed, re-named, and re-personalized on multiple hardware platforms, including virtual
2160 hardware.
- 2161 • **Portability class 2.** Runs on a specific family of virtual hardware. This is typically due to lack of
2162 driver support by the installed guest software.
- 2163 • **Portability class 3.** Only runs on a particular virtualization product and/or CPU architecture
2164 and/or virtual hardware selection. This is typically due to the OVF containing suspended virtual
2165 machines or snapshots of powered on virtual machines, including the current run-time state of
2166 the CPU and real or emulated devices. Such a state ties the OVF to a very specific virtualization
2167 and hardware platform.

2168 For use within an organization, class 2 or class 3 compatibility may be good enough because the OVF
2169 package is distributed within a controlled environment where specific purchasing decisions of hardware or
2170 virtualization platforms can ensure consistency of the underlying feature set for the OVF. A simple export
2171 of a virtual machine creates an OVF with class 3 or class 2 portability (tied to a specific set of virtual

2172 hardware); however, it is easy to extend the metaphor to support the export of class 1 portability, for
2173 example through the use of utilities such as “sysprep” for Windows.

2174 For commercial appliances independently created and distributed by ISVs, class 1 portability is desirable.
2175 Indeed, class 1 portability ensures that the appliance is readily available for the broadest possible
2176 customer base both for evaluation and production. Toolkits are used to create certified “known good”
2177 class 1 packages of the appliance for broad distribution and installation on multiple virtual platforms, or
2178 class 2 portability packages if the appliance is to be consumed within the context of a narrower set of
2179 virtual hardware, such as within a particular development group in an enterprise.

2180 The OVF virtual hardware description is designed to support class 1 through class 3 portability. For class
2181 1 portability, it is possible to include only very general descriptions of hardware requirements, or to
2182 specify multiple alternative virtual hardware descriptions. The appliance provider is in full control of how
2183 flexible or restrictive the virtual hardware specification is made. A narrow specification can be used to
2184 constrain an appliance to run on only known-good virtual hardware, while limiting its portability somewhat.
2185 A broad specification makes the appliance useful across as wide a set of virtual hardware as possible.
2186 This ensures that customers have the best possible user experience, which is one of the main
2187 requirements for the success of the virtual appliance concept.

ANNEX A (informative)

Single virtual system example

2188
2189
2190
2191

2192 Most of the descriptor is boilerplate. It starts out by describing the set of files in addition to the descriptor
2193 itself. In this case, there is a single file (`vmdisk1.vmdk`). It then describes the set of virtual disks and the
2194 set of networks used by the appliance. Each file, disk, and network resource is given a unique identifier.
2195 These are all in separate namespaces, but the best practice is to use distinct names.

2196 The content of the example OVF is a single virtual machine. The content contains five sections.

2197 The following listing shows a complete OVF descriptor for a typical single virtual machine appliance:

```
2198 <?xml version="1.0" encoding="UTF-8"?>
2199 <Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2200     xmlns="http://schemas.dmtf.org/ovf/1/envelope"
2201     xmlns:ovf="http://schemas.dmtf.org/ovf/1/envelope"
2202     xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2203     schema/2/CIM_VirtualSystemSettingData"
2204     xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2205     schema/2/CIM_ResourceAllocationSettingData">
2206
2207     <!-- References to all external files -->
2208     <References>
2209         <File ovf:id="file1" ovf:href="vmdisk1.vmdk" ovf:size="180114671"/>
2210     </References>
2211     <!-- Describes meta-information for all virtual disks in the package -->
2212     <DiskSection>
2213         <Info>Describes the set of virtual disks</Info>
2214         <Disk ovf:diskId="vmdisk1" ovf:fileRef="file1" ovf:capacity="4294967296"
2215         ovf:format="http://www.vmware.com/interfaces/specifications/vmdk.html#sparse"/>
2216     </DiskSection>
2217     <!-- Describes all networks used in the package -->
2218     <NetworkSection>
2219         <Info>List of logical networks used in the package</Info>
2220         <Network ovf:name="VM Network">
2221             <Description>The network that the services are available on</Description>
2222         </Network>
2223     </NetworkSection>
2224     <VirtualSystem ovf:id="vm">
2225         <Info>Describes a virtual machine</Info>
2226         <Name>Virtual Appliance One</Name>
2227         <ProductSection>
2228             <Info>Describes product information for the appliance</Info>
2229             <Product>The Great Appliance</Product>
2230             <Vendor>Some Great Corporation</Vendor>
2231             <Version>13.00</Version>
2232             <FullVersion>13.00-b5</FullVersion>
2233         </ProductSection>
2234
```

```

2235 <ProductUrl>http://www.somegreatcorporation.com/greatappliance</ProductUrl>
2236     <VendorUrl>http://www.somegreatcorporation.com/</VendorUrl>
2237     <Property ovf:key="adminEmail" ovf:type="string">
2238         <Description>Email address of administrator</Description>
2239     </Property>
2240     <Property ovf:key="appIp" ovf:type="string"
2241     ovf:defaultValue="192.168.0.10">
2242         <Description>The IP address of this appliance</Description>
2243     </Property>
2244 </ProductSection>
2245 <AnnotationSection ovf:required="false">
2246     <Info>A random annotation on this service. It can be ignored</Info>
2247     <Annotation>Contact customer support if you have any problems</Annotation>
2248 </AnnotationSection>
2249 <EulaSection>
2250     <Info>License information for the appliance</Info>
2251     <License>Insert your favorite license here</License>
2252 </EulaSection>
2253 <VirtualHardwareSection>
2254     <Info>256MB, 1 CPU, 1 disk, 1 nic</Info>
2255     <Item>
2256         <rasd:Description>Number of virtual CPUs</rasd:Description>
2257         <rasd:ElementName>1 virtual CPU</rasd:ElementName>
2258         <rasd:InstanceID>1</rasd:InstanceID>
2259         <rasd:ResourceType>3</rasd:ResourceType>
2260         <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2261     </Item>
2262     <Item>
2263         <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
2264         <rasd:Description>Memory Size</rasd:Description>
2265         <rasd:ElementName>256 MB of memory</rasd:ElementName>
2266         <rasd:InstanceID>2</rasd:InstanceID>
2267         <rasd:ResourceType>4</rasd:ResourceType>
2268         <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
2269     </Item>
2270     <Item>
2271         <rasd:AutomaticAllocation>true</rasd:AutomaticAllocation>
2272         <rasd:Connection>VM Network</rasd:Connection>
2273         <rasd:ElementName>Ethernet adapter on "VM Network"</rasd:ElementName>
2274         <rasd:InstanceID>4000</rasd:InstanceID>
2275         <rasd:ResourceType>10</rasd:ResourceType>
2276     </Item>
2277     <Item>
2278         <rasd:ElementName>Harddisk 1</rasd:ElementName>
2279         <rasd:HostResource>ovf:/disk/vmdisk1</rasd:HostResource>
2280         <rasd:InstanceID>22001</rasd:InstanceID>
2281         <rasd:ResourceType>17</rasd:ResourceType>
2282     </Item>
2283 </VirtualHardwareSection>
2284 <OperatingSystemSection ovf:id="58" ovf:required="false">

```

```
2285     <Info>Guest Operating System</Info>
2286     <Description>Windows 2000 Advanced Server</Description>
2287     </OperatingSystemSection>
2288 </VirtualSystem>
2289 </Envelope>
```

2290

**ANNEX B
(informative)**

Multitiered pet store example

2291
2292
2293
2294

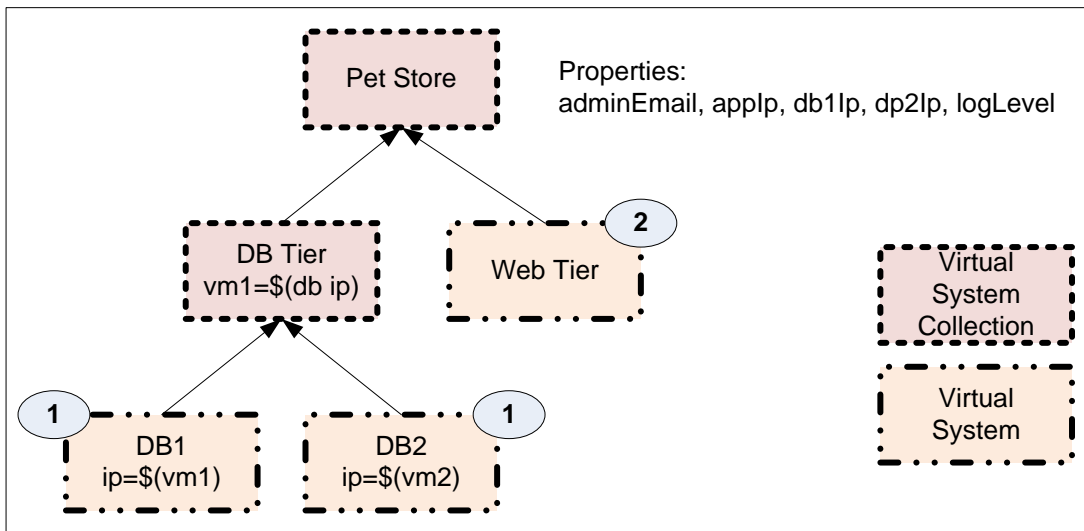
2295 The Pet Store OVF descriptor demonstrates several advanced OVF concepts:

- 2296 • Multi-VM packages - use of the VirtualMachineCollection entity subtype.
- 2297 • Composite service organization - use of a nested VirtualMachineCollection entity subtype.
- 2298 • Propagation of a user-defined deployment configuration.
- 2299 • Deployment time customization of the service using the OVF environment.
- 2300 • The use of virtual disk chains to minimize downloads.
- 2301 • Nesting of ProductSection elements for providing information about the installed software in
- 2302 an individual virtual machine.

2303 The example service is called Pet Store and consists of a front-end web-server and a database. The
2304 database server is itself a complex multitiered server consisting of two VMs for fault-tolerance.

B.1 Architecture and packaging

2306 The Pet Store OVF package consists of three virtual systems (WebTier, DB1, and DB2) and two virtual
2307 system collections (Pet Store and DBTier). Figure B-1 shows the structure of the OVF package as well as
2308 the properties and startup order of the virtual machines.



2309

Figure B-1 – Pet Store OVF package

2310

2311 The complete OVF descriptor is listed at the end of this document. The use of properties and disk layout
2312 of the OVF is discussed in more details in the following clauses.

B.2 Properties

2314 The Pet Store service has five user-configurable properties. These are the key control parameters for the
2315 service that need to be configured in order for it to start up correctly in the deployed environment. The
2316 properties are passed up to the guest software in the form of an OVF environment document. The guest

2317 software is written to read the OVF environment on startup, extract the values of the properties, and apply
 2318 them to the software configuration. Thus, the OVF descriptor reflects the properties that are handled by
 2319 the guest software.

2320 For this particular service, there are two different software configurations: one for the Web tier and one for
 2321 the Database tier. The properties supported in each software configuration are described in the following
 2322 tables.

2323 Table B-1 illustrates the properties for the Web Guest Software:

2324 **Table B-1 – Web tier configuration**

Property	Description
<i>applp</i>	IP address of the Web Server
<i>dblp</i>	IP address of the database server to which to connect
<i>adminEmail</i>	Email address for support
<i>logLevel</i>	Logging level

2325

2326 All properties defined on the immediate parent `VirtualSystemCollection` container are available to
 2327 a child `VirtualSystem` or `VirtualSystemCollection`. Thus, the OVF descriptor does not need to
 2328 contain an explicit `ProductSection` for each VM, as demonstrated for `WebVM`.

2329 Table B-2 illustrates the properties for the Database Guest Software:

2330 **Table B-2 – Database tier configuration**

Property	Description
<i>ip</i>	IP address of the virtual machine
<i>primaryAtBoot</i>	Whether the instance acts as the primary or secondary when booting
<i>ip2</i>	IP address of the twin database VM that acts as the hot-spare or primary
<i>log</i>	Logging level (called log here)

2331 The clustered database is organized as a virtual system collection itself with a specific set of properties
 2332 for configuration: `vm1`, `vm2`, and `log`. This organization separates the database implementation from the
 2333 rest of the software in the OVF package and allows virtual appliances (guest software + virtual machine
 2334 configurations) to be easily composed and thereby promotes reuse.

2335 The database software is an off-the-shelf software package and the vendor has chosen "com.mydb.db"
 2336 as the unique name for all the properties. This string can be seen in the OVF descriptor with the inclusion
 2337 of the `ovf:class` attribute on the `ProductSection`.

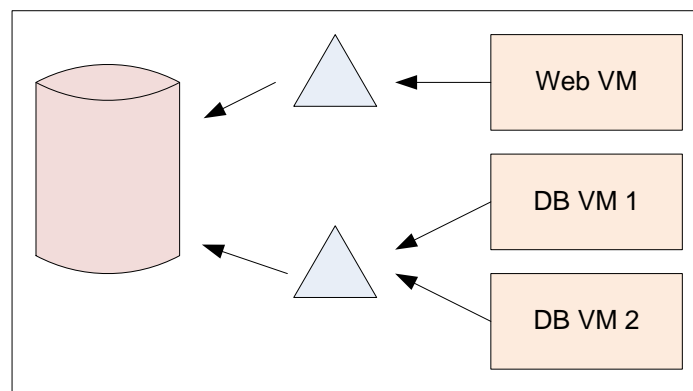
2338 The `${<name>}` property syntax is used to propagate values from the outer level into the inner nodes in
 2339 the OVF descriptor's entity hierarchy. This mechanism allows linking up different components without
 2340 having to pre-negotiate naming conventions or changing guest software. Only properties defined on the
 2341 immediate parent `VirtualSystemCollection` container are available to a child entity. Thus,
 2342 properties defined for the Petstore virtual system are not available to the DB1 virtual system. This ensures

2343 that the interface for a `VirtualSystemCollection` is encapsulated and well described in its parent
 2344 `VirtualSystemCollection`, which makes the software composable and easy to reuse.

2345 The OVF descriptor uses fixed non-user assignable properties to ensure that the two database virtual
 2346 machines boot up into different roles even though they are, initially, booting from the exact same software
 2347 image. The property named `com.mydb.db.primaryAtBoot` is specified with a fixed, non-user configurable
 2348 value but is different for the two images. The software inspects this value at boot time and customizes its
 2349 operation accordingly.

2350 B.3 Disk layout

2351 The Pet Store OVF package uses the ability to share disks and encode a delta disk hierarchy to minimize
 2352 the size and thereby the download time for the package. In this particular case, we only have two different
 2353 images (Database and Web), and if we further assume they are built on top of the same base OS
 2354 distribution, we can encode this configuration in the OVF descriptor as shown in Figure B-2.



2355

2356

Figure B-2 – Pet Store virtual disk layout

2357 Thus, while the package contains three distinct virtual machines, the total download size is significantly
 2358 smaller. In fact, only one full VM and then two relative small deltas need to be downloaded.

2359 The physical layout of the virtual disks on the deployment system is independent of the disk structure in
 2360 the OVF package. The OVF package describes the size of the virtual disk and the content (i.e., bits that
 2361 needs to be on the disk). It also specifies that each virtual machine gets independent disks. Thus, a
 2362 virtualization platform could install the above package as three VMs with three independent flat disks, or it
 2363 could chose to replicate the above organization, or something else, as long as each virtual machine sees
 2364 a disk with the content described on initial boot and that changes written by one virtual machine do not
 2365 affect the others.

2366 B.4 Pet Store OVF descriptor

```

2367 <?xml version="1.0" encoding="UTF-8"?>
2368 <Envelope
2369   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2370   xmlns="http://schemas.dmtf.org/ovf/envelope/1"
2371   xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1"
2372   xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2373 schema/2/CIM_VirtualSystemSettingData"
2374   xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2375 schema/2/CIM_ResourceAllocationSettingData">
2376   <!-- References to all external files -->
2377   <References>
2378     <File ovf:id="base" ovf:href="base.vmdk" ovf:size="180114671"/>
2379     <File ovf:id="webdelta" ovf:href="webapp-delta.vmdk" ovf:size="123413"/>
  
```

```

2380     <File ovf:id="dbdelta" ovf:href="dbapp-delta.vmdk" ovf:size="343243"/>
2381 </References>
2382 <!-- Describes meta-information about all virtual disks in the package.
2383 This example is encoded as a delta-disk hierarchy.
2384 -->
2385 <DiskSection>
2386     <Info>Describes the set of virtual disks</Info>
2387     <Disk ovf:diskId="base" ovf:fileRef="base" ovf:capacity="4294967296"
2388         ovf:populatedSize="1924967692"
2389 ovf:format="http://www.vmware.com/specifications/vmdk.html#streamOptimized"/>
2390     <Disk ovf:diskId="web" ovf:fileRef="webappdelta" ovf:parentRef="base"
2391         ovf:capacity="4294967296"
2392 ovf:format="http://www.vmware.com/specifications/vmdk.html#streamOptimized"/>
2393     <Disk ovf:diskId="db" ovf:fileRef="dbdelta" ovf:parentRef="base"
2394         ovf:capacity="4294967296"
2395 ovf:format="http://www.vmware.com/specifications/vmdk.html#streamOptimized"/>
2396 </DiskSection>
2397 <!-- Describes all networks used in the package -->
2400 <NetworkSection>
2401     <Info>List of logical networks used in the package</Info>
2402     <Network ovf:name="VM Network">
2403         <Description ovf:msgid="network.description">The network that the services
2404             are available on</Description>
2405     </Network>
2406 </NetworkSection>
2407 <!-- Deployment options for the packages -->
2409 <DeploymentOptionSection>
2410     <Info>List of deployment options available in the package</Info>
2411     <Configuration ovf:id="minimal">
2412         <Label ovf:msgid="minimal.label">Minimal</Label>
2413         <Description ovf:msgid="minimal.description">Deploy service with minimal
2414             resource use</Description>
2415     </Configuration>
2416     <Configuration ovf:id="standard" ovf:default="true">
2417         <Label ovf:msgid="standard.label">Standard</Label>
2418         <Description ovf:msgid="standard.description">Deploy service with standard
2419             resource use</Description>
2420     </Configuration>
2421 </DeploymentOptionSection>
2422 <!-- PetStore Virtual System Collection -->
2423 <VirtualSystemCollection ovf:id="PetStore">
2424     <Info>The packaging of the PetStoreService multitier application</Info>
2425     <Name>PetStore Service</Name>
2426     <!-- Overall information about the product -->
2427     <ProductSection>
2428         <Info>Describes product information for the service</Info>
2429         <Product>PetStore Web Portal</Product>
2430         <Vendor>Some Random Organization</Vendor>
2431         <Version>4.5</Version>
2432         <FullVersion>4.5-b4523</FullVersion>
2433         <ProductUrl>http://www.vmware.com/go/ovf</ProductUrl>
2434         <VendorUrl>http://www.vmware.com/</VendorUrl>
2435         <Category ovf:msgid="category.email">Email properties</Category>
2436         <Property ovf:key="adminEmail" ovf:type="string"
2437 ovf:userConfigurable="true">
2438             <Label ovf:msgid="property.email.label">Admin email</Label>
2439             <Description ovf:msgid="property.email.description">Email address of
2440                 service administrator</Description>
2441         </Property>
2442         <Category ovf:msgid="category.network">Network properties</Category>
2443         <Property ovf:key="appIp" ovf:type="string"

```



```

2444         ovf:userConfigurable="true">
2445         <Label ovf:msgid="property.appIp.label">IP</Label>
2446         <Description ovf:msgid="property.appIp.description">IP address of the
2447         service</Description>
2448     </Property>
2449     <Property ovf:key="dbIp" ovf:type="string" ovf:userConfigurable="true">
2450         <Label ovf:msgid="property.dpip.label">IP for DB</Label>
2451         <Description ovf:msgid="property.dpip.description">Primary IP address
2452 of
2453         the database</Description>
2454     </Property>
2455     <Property ovf:key="db2Ip" ovf:type="string"
2456         ovf:userConfigurable="true">
2457         <Label ovf:msgid="property.dpip2.label">IP for DB2</Label>
2458         <Description ovf:msgid="property.dpip2.description">A secondary IP
2459         address for the database</Description>
2460     </Property>
2461     <Category ovf:msgid="category.logging">Logging properties</Category>
2462     <Property ovf:key="logLevel" ovf:type="string" ovf:value="normal"
2463         ovf:userConfigurable="true">
2464         <Label ovf:msgid="property.loglevel.label">Loglevel</Label>
2465         <Description ovf:msgid="property.loglevel.description">Logging level
2466 for
2467         the service</Description>
2468         <Value ovf:value="low" ovf:configuration="minimal"/>
2469     </Property>
2470 </ProductSection>
2471 <AnnotationSection ovf:required="false">
2472     <Info>A annotation on this service</Info>
2473     <Annotation ovf:msgid="annotation.annotation">Contact customer support for
2474     any urgent issues</Annotation>
2475 </AnnotationSection>
2476 <ResourceAllocationSection ovf:required="false">
2477     <Info>Defines minimum reservations for CPU and memory</Info>
2478     <Item>
2479         <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
2480         <rasd:ElementName>512 MB reservation</rasd:ElementName>
2481         <rasd:InstanceID>0</rasd:InstanceID>
2482         <rasd:Reservation>512</rasd:Reservation>
2483         <rasd:ResourceType>4</rasd:ResourceType>
2484     </Item>
2485     <Item ovf:configuration="minimal">
2486         <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
2487         <rasd:ElementName>384 MB reservation</rasd:ElementName>
2488         <rasd:InstanceID>0</rasd:InstanceID>
2489         <rasd:Reservation>384</rasd:Reservation>
2490         <rasd:ResourceType>4</rasd:ResourceType>
2491     </Item>
2492     <Item>
2493         <rasd:AllocationUnits>MHz</rasd:AllocationUnits>
2494         <rasd:ElementName>1000 MHz reservation</rasd:ElementName>
2495         <rasd:InstanceID>1</rasd:InstanceID>
2496         <rasd:Reservation>500</rasd:Reservation>
2497         <rasd:ResourceType>3</rasd:ResourceType>
2498     </Item>
2499     <Item ovf:bound="min">
2500         <rasd:AllocationUnits>MHz</rasd:AllocationUnits>
2501         <rasd:ElementName>500 MHz reservation</rasd:ElementName>
2502         <rasd:InstanceID>1</rasd:InstanceID>
2503         <rasd:Reservation>500</rasd:Reservation>
2504         <rasd:ResourceType>3</rasd:ResourceType>
2505     </Item>
2506     <Item ovf:bound="max">
2507         <rasd:AllocationUnits>MHz</rasd:AllocationUnits>

```

```

2508         <rasd:ElementName>1500 MHz reservation</rasd:ElementName>
2509         <rasd:InstanceID>1</rasd:InstanceID>
2510         <rasd:Reservation>1500</rasd:Reservation>
2511         <rasd:ResourceType>3</rasd:ResourceType>
2512     </Item>
2513 </ResourceAllocationSection>
2514 <StartupSection>
2515     <Info>Specifies how the composite service is powered-on and off</Info>
2516     <Item ovf:id="DBTier" ovf:order="1" ovf:startDelay="120"
2517         ovf:startAction="powerOn" ovf:waitingForGuest="true"
2518 ovf:stopDelay="120"
2519         ovf:stopAction="guestShutdown"/>
2520     <Item ovf:id="WebTier" ovf:order="2" ovf:startDelay="120"
2521         ovf:startAction="powerOn" ovf:waitingForGuest="true"
2522 ovf:stopDelay="120"
2523         ovf:stopAction="guestShutdown"/>
2524 </StartupSection>
2525 <VirtualSystem ovf:id="WebTier">
2526     <Info>The virtual machine containing the WebServer application</Info>
2527     <ProductSection>
2528         <Info>Describes the product information</Info>
2529         <Product>Apache Webserver</Product>
2530         <Vendor>Apache Software Foundation</Vendor>
2531         <Version>6.5</Version>
2532         <FullVersion>6.5-b2432</FullVersion>
2533     </ProductSection>
2534     <OperatingSystemSection ovf:id="97">
2535         <Info>Guest Operating System</Info>
2536         <Description>Linux 2.4.x</Description>
2537     </OperatingSystemSection>
2538     <VirtualHardwareSection>
2539         <Info>256 MB, 1 CPU, 1 disk, 1 nic virtual machine</Info>
2540         <System>
2541             <vssd:ElementName>Virtual Hardware Family</vssd:ElementName>
2542             <vssd:InstanceID>0</vssd:InstanceID>
2543             <vssd:VirtualSystemType>vmx-04</vssd:VirtualSystemType>
2544         </System>
2545         <Item>
2546             <rasd:Description>Number of virtual CPUs</rasd:Description>
2547             <rasd:ElementName>1 virtual CPU</rasd:ElementName>
2548             <rasd:InstanceID>1</rasd:InstanceID>
2549             <rasd:ResourceType>3</rasd:ResourceType>
2550             <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2551         </Item>
2552         <Item>
2553             <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
2554             <rasd:Description>Memory Size</rasd:Description>
2555             <rasd:ElementName>256 MB of memory</rasd:ElementName>
2556             <rasd:InstanceID>2</rasd:InstanceID>
2557             <rasd:ResourceType>4</rasd:ResourceType>
2558             <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
2559         </Item>
2560         <Item>
2561             <rasd:AutomaticAllocation>true</rasd:AutomaticAllocation>
2562             <rasd:Connection>VM Network</rasd:Connection>
2563             <rasd:ElementName>Ethernet adapter on "VM
2564 Network"</rasd:ElementName>
2565             <rasd:InstanceID>3</rasd:InstanceID>
2566             <rasd:ResourceSubType>PCNet32</rasd:ResourceSubType>
2567             <rasd:ResourceType>10</rasd:ResourceType>
2568         </Item>
2569         <Item>
2570             <rasd:AddressOnParent>1</rasd:AddressOnParent>
2571             <rasd:ElementName>SCSI Controller 0 - LSI Logic</rasd:ElementName>

```

```

2572         <rasd:InstanceID>1000</rasd:InstanceID>
2573         <rasd:ResourceSubType>LsiLogic</rasd:ResourceSubType>
2574         <rasd:ResourceType>6</rasd:ResourceType>
2575     </Item>
2576     <Item>
2577         <rasd:AddressOnParent>0</rasd:AddressOnParent>
2578         <rasd:ElementName>Harddisk 1</rasd:ElementName>
2579         <rasd:HostResource>ovf:/disk/web</rasd:HostResource>
2580         <rasd:InstanceID>22001</rasd:InstanceID>
2581         <rasd:Parent>1000</rasd:Parent>
2582         <rasd:ResourceType>17</rasd:ResourceType>
2583     </Item>
2584 </VirtualHardwareSection>
2585 </VirtualSystem>
2586 <!-- Database Tier -->
2587 <VirtualSystemCollection ovf:id="DBTier">
2588     <Info>Describes a clustered database instance</Info>
2589     <ProductSection ovf:class="com.mydb.db">
2590         <Info>Product Information</Info>
2591         <Product>Somebody Clustered SQL Server</Product>
2592         <Vendor>TBD</Vendor>
2593         <Version>2.5</Version>
2594         <FullVersion>2.5-b1234</FullVersion>
2595         <Property ovf:key="vm1" ovf:value="{dbIp}" ovf:type="string"/>
2596         <Property ovf:key="vm2" ovf:value="{db2Ip}" ovf:type="string"/>
2597         <Property ovf:key="log" ovf:value="{logLevel}" ovf:type="string"/>
2598     </ProductSection>
2599     <StartupSection>
2600         <Info>Specifies how the composite service is powered-on and off</Info>
2601         <Item ovf:id="DB1" ovf:order="1" ovf:startDelay="120"
2602             ovf:startAction="powerOn" ovf:waitingForGuest="true"
2603             ovf:stopDelay="120" ovf:stopAction="guestShutdown"/>
2604         <Item ovf:id="DB2" ovf:order="2" ovf:startDelay="120"
2605             ovf:startAction="powerOn" ovf:waitingForGuest="true"
2606             ovf:stopDelay="120" ovf:stopAction="guestShutdown"/>
2607     </StartupSection>
2608     <!-- DB VM 1 -->
2609     <VirtualSystem ovf:id="DB1">
2610         <Info>Describes a virtual machine with the database image
2611 installed</Info>
2612         <Name>Database Instance I</Name>
2613         <ProductSection ovf:class="com.mydb.db">
2614             <Info>Specifies the OVF properties available in the OVF
2615 environment</Info>
2616             <Property ovf:key="ip" ovf:value="{vm1}" ovf:type="string"/>
2617             <Property ovf:key="ip2" ovf:value="{vm2}" ovf:type="string"/>
2618             <Property ovf:key="primaryAtBoot" ovf:value="yes"
2619 ovf:type="string"/>
2620         </ProductSection>
2621         <VirtualHardwareSection>
2622             <Info>256 MB, 1 CPU, 1 disk, 1 nic virtual machine</Info>
2623             <System>
2624                 <vssd:ElementName>Virtual Hardware Family</vssd:ElementName>
2625                 <vssd:InstanceID>0</vssd:InstanceID>
2626                 <vssd:VirtualSystemType>vmx-04</vssd:VirtualSystemType>
2627             </System>
2628             <Item>
2629                 <rasd:Description>Number of virtual CPUs</rasd:Description>
2630                 <rasd:ElementName>1 virtual CPU</rasd:ElementName>
2631                 <rasd:InstanceID>1</rasd:InstanceID>
2632                 <rasd:ResourceType>3</rasd:ResourceType>
2633                 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2634             </Item>
2635         </VirtualSystem>

```

```

2636     <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
2637     <rasd:Description>Memory Size</rasd:Description>
2638     <rasd:ElementName>256 MB of memory</rasd:ElementName>
2639     <rasd:InstanceID>2</rasd:InstanceID>
2640     <rasd:ResourceType>4</rasd:ResourceType>
2641     <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
2642   </Item>
2643   <Item>
2644     <rasd:AutomaticAllocation>true</rasd:AutomaticAllocation>
2645     <rasd:Connection>VM Network</rasd:Connection>
2646     <rasd:ElementName>Ethernet adapter on "VM
2647 Network"</rasd:ElementName>
2648     <rasd:InstanceID>3</rasd:InstanceID>
2649     <rasd:ResourceSubType>PCNet32</rasd:ResourceSubType>
2650     <rasd:ResourceType>10</rasd:ResourceType>
2651   </Item>
2652   <Item>
2653     <rasd:AddressOnParent>1</rasd:AddressOnParent>
2654     <rasd:ElementName>SCSI Controller 0 - LSI
2655 Logic</rasd:ElementName>
2656     <rasd:InstanceID>1000</rasd:InstanceID>
2657     <rasd:ResourceSubType>LsiLogic</rasd:ResourceSubType>
2658     <rasd:ResourceType>6</rasd:ResourceType>
2659   </Item>
2660   <Item>
2661     <rasd:AddressOnParent>0</rasd:AddressOnParent>
2662     <rasd:ElementName>Harddisk 1</rasd:ElementName>
2663     <rasd:HostResource>ovf:/disk/db</rasd:HostResource>
2664     <rasd:InstanceID>22001</rasd:InstanceID>
2665     <rasd:Parent>1000</rasd:Parent>
2666     <rasd:ResourceType>17</rasd:ResourceType>
2667   </Item>
2668 </VirtualHardwareSection>
2669 <OperatingSystemSection ovf:id="97">
2670   <Info>Guest Operating System</Info>
2671   <Description>Linux 2.4.x</Description>
2672 </OperatingSystemSection>
2673 </VirtualSystem>
2674 <!-- DB VM 2 -->
2675 <VirtualSystem ovf:id="DB2">
2676   <Info>Describes a virtual machine with the database image
2677 installed</Info>
2678   <Name>Database Instance II</Name>
2679   <ProductSection ovf:class="com.mydb.db">
2680     <Info>Specifies the OVF properties available in the OVF
2681 environment</Info>
2682     <Property ovf:key="ip" ovf:value="{vm2}" ovf:type="string"/>
2683     <Property ovf:key="ip2" ovf:value="{vm1}" ovf:type="string"/>
2684     <Property ovf:key="primaryAtBoot" ovf:value="no"
2685 ovf:type="string"/>
2686   </ProductSection>
2687   <VirtualHardwareSection>
2688     <Info>256 MB, 1 CPU, 1 disk, 1 nic virtual machine</Info>
2689     <System>
2690       <vssd:ElementName>Virtual Hardware Family</vssd:ElementName>
2691       <vssd:InstanceID>0</vssd:InstanceID>
2692       <vssd:VirtualSystemType>vmx-04</vssd:VirtualSystemType>
2693     </System>
2694     <Item>
2695       <rasd:Description>Number of virtual CPUs</rasd:Description>
2696       <rasd:ElementName>1 virtual CPU</rasd:ElementName>
2697       <rasd:InstanceID>1</rasd:InstanceID>
2698       <rasd:ResourceType>3</rasd:ResourceType>
2699       <rasd:VirtualQuantity>1</rasd:VirtualQuantity>

```

```

2700         </Item>
2701     <Item>
2702         <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
2703         <rasd:Description>Memory Size</rasd:Description>
2704         <rasd:ElementName>256 MB of memory</rasd:ElementName>
2705         <rasd:InstanceID>2</rasd:InstanceID>
2706         <rasd:ResourceType>4</rasd:ResourceType>
2707         <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
2708     </Item>
2709     <Item>
2710         <rasd:AutomaticAllocation>true</rasd:AutomaticAllocation>
2711         <rasd:Connection>VM Network</rasd:Connection>
2712         <rasd:ElementName>Ethernet adapter on "VM
2713 Network"</rasd:ElementName>
2714         <rasd:InstanceID>3</rasd:InstanceID>
2715         <rasd:ResourceSubType>PCNet32</rasd:ResourceSubType>
2716         <rasd:ResourceType>10</rasd:ResourceType>
2717     </Item>
2718     <Item>
2719         <rasd:AddressOnParent>1</rasd:AddressOnParent>
2720         <rasd:ElementName>SCSI Controller 0 - LSI
2721 Logic</rasd:ElementName>
2722         <rasd:InstanceID>1000</rasd:InstanceID>
2723         <rasd:ResourceSubType>LsiLogic</rasd:ResourceSubType>
2724         <rasd:ResourceType>6</rasd:ResourceType>
2725     </Item>
2726     <Item>
2727         <rasd:AddressOnParent>0</rasd:AddressOnParent>
2728         <rasd:ElementName>Harddisk 1</rasd:ElementName>
2729         <rasd:HostResource>ovf:/disk/db</rasd:HostResource>
2730         <rasd:InstanceID>22001</rasd:InstanceID>
2731         <rasd:Parent>1000</rasd:Parent>
2732         <rasd:ResourceType>17</rasd:ResourceType>
2733     </Item>
2734 </VirtualHardwareSection>
2735 <OperatingSystemSection ovf:id="97">
2736     <Info>Guest Operating System</Info>
2737     <Description>Linux 2.4.x</Description>
2738 </OperatingSystemSection>
2739 </VirtualSystem>
2740 </VirtualSystemCollection>
2741 </VirtualSystemCollection>
2742 <!-- External I18N bundles -->
2743 <Strings xml:lang="de-DE" ovf:fileRef="de-DE-bundle.xml"/>
2744 <!-- Embedded I18N bundles -->
2745 <Strings xml:lang="da-DA">
2746     <Msg ovf:msgid="network.description">Netværket servicen skal være tilgængelig
2747 på</Msg>
2748     <Msg ovf:msgid="annotation.annotation">Kontakt kundeservice i tilfælde af
2749     kritiske problemer</Msg>
2750     <Msg ovf:msgid="property.email.description">Email adresse for
2751 administrator</Msg>
2752     <Msg ovf:msgid="property.appIp.description">IP adresse for service</Msg>
2753     <Msg ovf:msgid="property.dpIp">Primær IP adresse for database</Msg>
2754     <Msg ovf:msgid="property.dpIp2.description">Sekundær IP adresse for
2755 database</Msg>
2756     <Msg ovf:msgid="property.loglevel.description">Logningsniveau for
2757 service</Msg>
2758     <Msg ovf:msgid="minimal.label">Minimal</Msg>
2759     <Msg ovf:msgid="minimal.description">Installer service med minimal brug af
2760     resourcer</Msg>
2761     <Msg ovf:msgid="standard.label">Normal</Msg>
2762     <Msg ovf:msgid="standard.description">Installer service med normal brug af
2763     resourcer</Msg>

```

```
2764     </Strings>
2765 </Envelope>
```

2766 B.5 Complete OVF environment

2767 The following example lists the OVF environments that are seen by the WebTier and DB1 virtual
2768 machines. (DB2 is virtually identical to the one for DB1 and is omitted.)

2769 OVF environment for the WebTier virtual machine:

```
2770 <?xml version="1.0" encoding="UTF-8"?>
2771 <Environment
2772     xmlns="http://schemas.dmtf.org/ovf/environment/1"
2773     xmlns:ovfenv="http://schemas.dmtf.org/ovf/environment/1"
2774     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2775     ovfenv:id="WebTier">
2776
2777     <!-- Information about hypervisor platform -->
2778     <PlatformSection>
2779         <Kind>ESX Server</Kind>
2780         <Version>3.0.1</Version>
2781         <Vendor>VMware, Inc.</Vendor>
2782         <Locale>en_US</Locale>
2783     </PlatformSection>
2784
2785     <!-- Properties defined for this virtual machine -->
2786     <PropertySection>
2787         <Property ovfenv:key="adminEmail" ovfenv:value="ovf-admin@vmware.com"/>
2788         <Property ovfenv:key="appIp" ovfenv:value="10.20.132.101"/>
2789         <Property ovfenv:key="dbIp" ovfenv:value="10.20.132.102"/>
2790         <Property ovfenv:key="db2Ip" ovfenv:value="10.20.132.103"/>
2791         <Property ovfenv:key="logLevel" ovfenv:value="warning"/>
2792     </PropertySection>
2793
2794     <Entity ovfenv:id="DBTier">
2795         <PropertySection>
2796             <Property ovfenv:key="adminEmail" ovfenv:value="ovf-admin@vmware.com"/>
2797             <Property ovfenv:key="appIp" ovfenv:value="10.20.132.101"/>
2798             <Property ovfenv:key="dbIp" ovfenv:value="10.20.132.102"/>
2799             <Property ovfenv:key="db2Ip" ovfenv:value="10.20.132.103"/>
2800             <Property ovfenv:key="logLevel" ovfenv:value="warning"/>
2801             <Property ovfenv:key="com.mydb.db.vm1" ovfenv:value="10.20.132.102"/>
2802             <Property ovfenv:key="com.mydb.db.vm2" ovfenv:value="10.20.132.103"/>
2803             <Property ovfenv:key="com.mydb.db.log" ovfenv:value="warning"/>
2804         </PropertySection>
2805     </Entity>
2806 </Environment>
```

2807 OVF environment for the DB1 virtual machine:

```
2808 <?xml version="1.0" encoding="UTF-8"?>
2809 <Environment
2810     xmlns="http://schemas.dmtf.org/ovf/environment/1"
2811     xmlns:ovfenv="http://schemas.dmtf.org/ovf/environment/1"
2812     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2813     ovfenv:id="DB1">
2814
2815     <!-- Information about hypervisor platform -->
2816     <PlatformSection>
2817         <Kind>ESX Server</Kind>
2818         <Version>3.0.1</Version>
2819         <Vendor>VMware, Inc.</Vendor>
```

```
2820     <Locale>en_US</Locale>
2821 </PlatformSection>
2822
2823 <!-- Properties defined for this virtual machine -->
2824 <PropertySection>
2825     <Property ovfenv:key="com.mydb.db.vm1" ovfenv:value="10.20.132.102"/>
2826     <Property ovfenv:key="com.mydb.db.vm2" ovfenv:value="10.20.132.103"/>
2827     <Property ovfenv:key="com.mydb.db.log" ovfenv:value="warning"/>
2828     <Property ovfenv:key="com.mydb.db.ip" ovfenv:value="10.20.132.102"/>
2829     <Property ovfenv:key="com.mydb.db.ip2" ovfenv:value="10.20.132.103"/>
2830     <Property ovfenv:key="com.mydb.db.primaryAtBoot" ovfenv:value="yes"/>
2831 </PropertySection>
2832
2833 <Entity ovfenv:id="DB2">
2834     <PropertySection>
2835         <Property ovfenv:key="com.mydb.db.vm1" ovfenv:value="10.20.132.102"/>
2836         <Property ovfenv:key="com.mydb.db.vm2" ovfenv:value="10.20.132.103"/>
2837         <Property ovfenv:key="com.mydb.db.log" ovfenv:value="warning"/>
2838         <Property ovfenv:key="com.mydb.db.ip" ovfenv:value="10.20.132.103"/>
2839         <Property ovfenv:key="com.mydb.db.ip2" ovfenv:value="10.20.132.102"/>
2840         <Property ovfenv:key="com.mydb.db.primaryAtBoot" ovfenv:value="no"/>
2841     </PropertySection>
2842 </Entity>
2843 </Environment>
2844
```

ANNEX C (informative)

Single virtual system LAMP stack example

2845
2846
2847
2848

2849 In this example, we provide two concrete examples of how an OVF descriptor for a LAMP virtual
2850 appliance could look. We show both a single-VM LAMP virtual appliance and a multi-VM LAMP virtual
2851 appliance. LAMP is an abbreviation for a service built by using the Linux operating system, Apache web
2852 server, MySQL database, and the PHP web development software packages.

2853 This examples show how the `ProductSection` can be used to specify both operating system and
2854 application-level deployment parameters. For example, these parameters can be used to optimize the
2855 performance of a service when deployed into a particular environment. The descriptors are complete, but
2856 otherwise kept minimal; for example, there are no EULA sections.

2857 C.1 Deployment-time customization

2858 A part of the deployment phase of an OVF package is to provide customization parameters. The
2859 customization parameters are specified in the OVF descriptor and are provided to the guest software
2860 using the OVF environment. This deployment time customization is in addition to the virtual machine level
2861 parameters, which includes virtual switch connectivity and physical storage location.

2862 For a LAMP-based virtual appliance, the deployment time customization includes the IP address and port
2863 number of the service, network information, such as gateway and subnet, and also parameters, so the
2864 performance can be optimized for a given deployment. The properties that are exposed to the deployer
2865 vary from vendor to vendor and service to service. In the example descriptors, the following set of
2866 properties described in Table C-1 are used for the four different LAMP components:

2867

Table C-1 – LAMP configuration

Product	Property	Description
Linux	<i>hostname</i>	Network identity of the application, including IP address
	<i>ip</i>	
	<i>subnet</i>	
	<i>gateway</i>	
	<i>dns</i>	
Linux	<i>netCoreRmemMax</i>	Parameters to optimize the transfer rate of the IP stack
	<i>netCoreWmemMax</i>	

Product	Property	Description
Apache	httpPort	Port numbers for web server
	httpsPort	
	startThreads	Parameters to optimize the performance of the web server
	minSpareThreads	
	maxSpareThreads	
	maxClients	
MySQL	queryCacheSize	Parameters to optimize the performance of database
	maxConnections	
	waitTimeout	
PHP	sessionTimeout	Parameters to customize the behavior of the PHP engine, including how sessions timeout and number of sessions
	concurrentSessions	
	memoryLimit	

2868 The parameters in *italic* are required configurations from the user. Otherwise, they have reasonable
 2869 defaults, so the user does not necessarily need to provide a value.

2870 The customization parameters for each software product are encapsulated in separate product sections.
 2871 For example, for the Apache web server the following section is used:

```

2872 <ProductSection ovf:class="org.apache.httpd">
2873   <Info>Product customization for the installed Apache Web Server</Info>
2874   <Product>Apache Distribution Y</Product>
2875   <Version>2.6.6</Version>
2876   <Property ovf:key="httpPort" ovf:type="uint16" ovf:value="80"
2877     ovf:userConfigurable="true">
2878     <Description>Port number for HTTP requests</Description>
2879   </Property>
2880   <Property ovf:key="httpsPort" ovf:type="uint16" ovf:value="443"
2881     ovf:userConfigurable="true">
2882     <Description>Port number for HTTPS requests</Description>
2883   </Property>
2884   <Property ovf:key="startThreads" ovf:type="uint16" ovf:value="50"
2885     ovf:userConfigurable="true">
2886     <Description>Number of threads created on startup. </Description>
2887   </Property>
2888   <Property ovf:key="minSpareThreads" ovf:type="uint16" ovf:value="15"
2889     ovf:userConfigurable="true">
2890     <Description> Minimum number of idle threads to handle request
2891 spikes.</Description>
2892   </Property>
2893   <Property ovf:key="maxSpareThreads" ovf:type="uint16" ovf:value="30"
2894     ovf:userConfigurable="true">
2895     <Description>Maximum number of idle threads </Description>
2896   </Property>
2897   <Property ovf:key="maxClients" ovf:type="uint16" ovf:value="256"
2898     ovf:userConfigurable="true">
2899     <Description>Limit the number of simultaneous requests that are served.
2900 </Description>
2901   </Property>
2902 </ProductSection>

```

2903 The `ovf:class="org.apache.httpd"` attribute specifies the prefix for the properties. Hence, the
 2904 Apache database is expected to look for the following properties in the OVF environment:

```

2905 <Environment
2906   ...
2907   <!-- Properties defined for this virtual machine -->
2908   <PropertySection>
2909     <Property ovfenv:name="org.apache.httpd.httpPort" ovfenv:value="80"/>
2910     <Property ovfenv:name="org.apache.httpd.httpsPort" ovfenv:value="443"/>
2911     <Property ovfenv:name="org.apache.httpd.startThreads" ovfenv:value="50"/>
2912     <Property ovfenv:name="org.apache.httpd.minSpareThreads" ovfenv:value="15"/>
2913     <Property ovfenv:name="org.apache.httpd.maxSpareThreads" ovfenv:value="30"/>
2914     <Property ovfenv:name="org.apache.httpd.maxClients" ovfenv:value="256"/>
2915     ...
2916   </PropertySection>
2917   ...
2918 </Environment>

```

2919 C.2 Simple LAMP OVF descriptor

2920 A complete OVF descriptor for a single VM virtual appliance with the LAMP stack is listed below:

```

2921 <?xml version="1.0" encoding="UTF-8"?>
2922 <Envelope
2923     xmlns="http://schemas.dmtf.org/ovf/envelope/1"
2924     xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1"
2925     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2926     xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2927 schema/2/CIM_VirtualSystemSettingData"
2928     xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2929 schema/2/CIM_ResourceAllocationSettingData"
2930     <!-- References to all external files -->
2931     <References>
2932         <File ovf:id="lamp" ovf:href="lamp.vmdk" ovf:size="180114671"/>
2933     </References>
2934     <!-- Describes meta-information about all virtual disks in the package. -->
2935     <DiskSection>
2936         <Info>List of the virtual disks used in the package</Info>
2937         <Disk ovf:diskId="lamp" ovf:fileRef="lamp" ovf:capacity="4294967296"
2938             ovf:populatedSize="1924967692"
2939             ovf:format="http://www.vmware.com/specifications/vmdk.html#streamOptimized"/>
2940     </DiskSection>
2941     <!-- Describes all networks used in the package -->
2942     <NetworkSection>
2943         <Info>Logical networks used in the package</Info>
2944         <Network ovf:name="VM Network">
2945             <Description>The network that the LAMP Service is available
2946                 on</Description>
2947         </Network>
2948     </NetworkSection>
2949     <VirtualSystem ovf:id="MyLampService">
2950         <Info>Single-VM Virtual appliance with LAMP stack</Info>
2951         <Name>LAMP Virtual Appliance</Name>
2952         <!-- Overall information about the product -->
2953         <ProductSection>
2954             <Info>Product information for the service</Info>
2955             <Product>Lamp Service</Product>
2956             <Version>1.0</Version>
2957             <FullVersion>1.0.0</FullVersion>
2958         </ProductSection>
2959         <!-- Linux component configuration parameters -->
2960         <ProductSection ovf:class="org.linuxdistx">
2961             <Info>Product customization for the installed Linux system</Info>
2962             <Product>Linux Distribution X</Product>
2963             <Version>2.6.3</Version>
2964             <Property ovf:key="hostname" ovf:type="string">
2965                 <Description>Specifies the hostname for the appliance</Description>
2966             </Property>
2967             <Property ovf:key="ip" ovf:type="string">
2968                 <Description>Specifies the IP address for the appliance</Description>
2969             </Property>
2970             <Property ovf:key="subnet" ovf:type="string">
2971                 <Description> Specifies the subnet to use on the deployed network
2972             </Description>
2973             </Property>
2974             <Property ovf:key="gateway" ovf:type="string">
2975                 <Description> Specifies the gateway on the deployed network
2976             </Description>
2977             </Property>
2978         </ProductSection>

```

```

2979     <Property ovf:key="dns" ovf:type="string">
2980         <Description> A comma separated list of DNS servers on the deployed
2981             network </Description>
2982     </Property>
2983     <Property ovf:key="netCoreRmemMaxMB" ovf:type="uint16" ovf:value="16"
2984         ovf:userConfigurable="true">
2985         <Description> Specify TCP read max buffer size in mega bytes. Default
2986 is
2987             16. </Description>
2988     </Property>
2989     <Property ovf:key="netCoreWmemMaxMB" ovf:type="uint16" ovf:value="16"
2990         ovf:userConfigurable="true">
2991 is
2992             <Description> Specify TCP write max buffer size in mega bytes. Default
2993             16. </Description>
2994     </Property>
2995 </ProductSection>
2996 <!-- Apache component configuration parameters -->
2997 <ProductSection ovf:class="org.apache.httpd">
2998     <Info>Product customization for the installed Apache Web Server</Info>
2999     <Product>Apache Distribution Y</Product>
3000     <Version>2.6.6</Version>
3001     <Property ovf:key="httpPort" ovf:type="uint16" ovf:value="80"
3002         ovf:userConfigurable="true">
3003         <Description>Port number for HTTP requests</Description>
3004     </Property>
3005     <Property ovf:key="httpsPort" ovf:type="uint16" ovf:value="443"
3006         ovf:userConfigurable="true">
3007         <Description>Port number for HTTPS requests</Description>
3008     </Property>
3009     <Property ovf:key="startThreads" ovf:type="uint16" ovf:value="50"
3010         ovf:userConfigurable="true">
3011         <Description>Number of threads created on startup. </Description>
3012     </Property>
3013     <Property ovf:key="minSpareThreads" ovf:type="uint16" ovf:value="15"
3014         ovf:userConfigurable="true">
3015         <Description> Minimum number of idle threads to handle request spikes.
3016 </Description>
3017     </Property>
3018     <Property ovf:key="maxSpareThreads" ovf:type="uint16" ovf:value="30"
3019         ovf:userConfigurable="true">
3020         <Description>Maximum number of idle threads </Description>
3021     </Property>
3022     <Property ovf:key="maxClients" ovf:type="uint16" ovf:value="256"
3023         ovf:userConfigurable="true">
3024         <Description>Limit the number of simultaneous requests that are
3025             served. </Description>
3026     </Property>
3027 </ProductSection>
3028 <!-- MySQL component configuration parameters -->
3029 <ProductSection ovf:class="org.mysql.db">
3030     <Info>Product customization for the installed MySql Database Server</Info>
3031     <Product>MySQL Distribution Z</Product>
3032     <Version>5.0</Version>
3033     <Property ovf:key="queryCacheSizeMB" ovf:type="uint16" ovf:value="32"
3034         ovf:userConfigurable="true">
3035         <Description>Buffer to cache repeated queries for faster access (in
3036             MB)</Description>
3037     </Property>
3038     <Property ovf:key="maxConnections" ovf:type="uint16" ovf:value="500"
3039         ovf:userConfigurable="true">
3040         <Description>The number of concurrent connections that can be
3041             served</Description>
3042     </Property>

```

```

3043     <Property ovf:key="waitTimeout" ovf:type="uint16" ovf:value="100"
3044         ovf:userConfigurable="true">
3045         <Description>Number of seconds to wait before timing out a connection
3046         </Description>
3047     </Property>
3048 </ProductSection>
3049 <!-- PHP component configuration parameters -->
3050 <ProductSection ovf:class="net.php">
3051     <Info>Product customization for the installed PHP component</Info>
3052 <Product>PHP Distribution U</Product>
3053 <Version>5.0</Version>
3054 <Property ovf:key="sessionTimeout" ovf:type="uint16" ovf:value="5"
3055     ovf:userConfigurable="true">
3056     <Description> How many minutes a session has to be idle before it is
3057     timed out </Description>
3058 </Property>
3059 <Property ovf:key="concurrentSessions" ovf:type="uint16" ovf:value="500"
3060     ovf:userConfigurable="true">
3061     <Description> The number of concurrent sessions that can be served
3062     </Description>
3063 </Property>
3064 <Property ovf:key="memoryLimit" ovf:type="uint16" ovf:value="32"
3065     ovf:userConfigurable="true">
3066     <Description> How much memory in megabytes a script can consume before
3067     being killed </Description>
3068 </Property>
3069 </ProductSection>
3070 <OperatingSystemSection ovf:id="99">
3071     <Info>Guest Operating System</Info>
3072     <Description>Linux 2.6.x</Description>
3073 </OperatingSystemSection>
3074 <VirtualHardwareSection>
3075     <Info>Virtual Hardware Requirements: 256MB, 1 CPU, 1 disk, 1 NIC</Info>
3076     <System>
3077         <vssd:ElementName>Virtual Hardware Family</vssd:ElementName>
3078         <vssd:InstanceID>0</vssd:InstanceID>
3079         <vssd:VirtualSystemType>vmx-04</vssd:VirtualSystemType>
3080     </System>
3081     <Item>
3082         <rasd:Description>Number of virtual CPUs</rasd:Description>
3083         <rasd:ElementName>1 virtual CPU</rasd:ElementName>
3084         <rasd:InstanceID>1</rasd:InstanceID>
3085         <rasd:ResourceType>3</rasd:ResourceType>
3086         <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
3087     </Item>
3088     <Item>
3089         <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
3090         <rasd:Description>Memory Size</rasd:Description>
3091         <rasd:ElementName>256 MB of memory</rasd:ElementName>
3092         <rasd:InstanceID>2</rasd:InstanceID>
3093         <rasd:ResourceType>4</rasd:ResourceType>
3094         <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
3095     </Item>
3096     <Item>
3097         <rasd:AutomaticAllocation>true</rasd:AutomaticAllocation>
3098         <rasd:Connection>VM Network</rasd:Connection>
3099         <rasd:ElementName>Ethernet adapter on "VM Network"</rasd:ElementName>
3100         <rasd:InstanceID>3</rasd:InstanceID>
3101         <rasd:ResourceType>10</rasd:ResourceType>
3102     </Item>
3103     <Item>
3104         <rasd:ElementName>SCSI Controller 0 - LSI Logic</rasd:ElementName>
3105         <rasd:InstanceID>4</rasd:InstanceID>
3106         <rasd:ResourceSubType>LsiLogic</rasd:ResourceSubType>

```

```
3107         <rasd:ResourceType>6</rasd:ResourceType>
3108     </Item>
3109     <Item>
3110         <rasd:ElementName>Harddisk 1</rasd:ElementName>
3111         <rasd:HostResource>ovf:/disk/lamp</rasd:HostResource>
3112         <rasd:InstanceID>5</rasd:InstanceID>
3113         <rasd:Parent>4</rasd:Parent>
3114         <rasd:ResourceType>17</rasd:ResourceType>
3115     </Item>
3116 </VirtualHardwareSection>
3117 </VirtualSystem>
3118 </Envelope>
```

3119

ANNEX D (informative)

Multiple virtual system LAMP stack example

This example is what an OVF descriptor for a LAMP virtual appliance could look like in a multi-VM LAMP virtual appliance. LAMP is an abbreviation for a service built using the Linux operating system, Apache web server, MySQL database, and the PHP web development software packages.

D.1 Two-tier LAMP OVF descriptor

In a two-tier LAMP stack, the application tier (Linux, Apache, PHP) and the database tier (Linux, MySQL server) are run as separate virtual machines for greater scalability.

The OVF format makes the process of how a service is implemented transparent to the user. In particular, the deployment experience when a user is installing a single-VM or a two-tier LAMP appliance is very similar. The only visible difference is that the user needs to supply two IP addresses and two DNS host names.

As compared to the single-VM descriptor, the following changes are made:

- All the user-configurable parameters are put in the `VirtualSystemCollection` entity. The `ProductSection` elements for Apache, MySQL, and PHP are unchanged from the single VM case.
- The Linux software in the two virtual machines needs to be configured slightly different (IP and hostname) while sharing most parameters. A new `ProductSection` is added to the `VirtualSystemCollection` to prompt the user, and the `${property}` expression is used to assign the values in each `VirtualSystem` entity.
- Disk chains are used to keep the download size comparable to that of a single VM appliance. Because the Linux installation is stored on a shared base disk, effectively only one copy of Linux needs to be downloaded.

The complete OVF descriptor is shown below:

```

3146 <?xml version="1.0" encoding="UTF-8"?>
3147 <Envelope
3148   xmlns="http://schemas.dmtf.org/ovf/envelope/1"
3149   xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1"
3150   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3151   xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-
3152 schema/2/CIM_VirtualSystemSettingData"
3153   xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
3154 schema/2/CIM_ResourceAllocationSettingData"
3155   <!-- References to all external files. -->
3156   <References>
3157     <File ovf:id="lamp-base" ovf:href="lampdb.vmdk" ovf:size="180114671"/>
3158     <File ovf:id="lamp-db" ovf:href="lampdb.vmdk" ovf:size="1801146"/>
3159     <File ovf:id="lamp-app" ovf:href="lampapp.vmdk" ovf:size="34311371"/>
3160   </References>
3161   <!-- Describes meta-information about all virtual disks in the package.
3162     This example is encoded as a delta-disk hierarchy.
3163   -->
3164   <DiskSection>
3165     <Info>List of the virtual disks used in the package</Info>
3166     <Disk ovf:diskId="lamp-base" ovf:fileRef="lamp-base" ovf:capacity="4294967296"
3167       ovf:populatedSize="1924967692"

```

```

3168
3169 ovf:format="http://www.vmware.com/specifications/vmdk.html#streamOptimized"/>
3170     <Disk ovf:diskId="lamp-db" ovf:fileRef="lamp-db" ovf:capacity="4294967296"
3171         ovf:populatedSize="19249672"
3172
3173 ovf:format="http://www.vmware.com/specifications/vmdk.html#streamOptimized"
3174     ovf:parentRef="lamp-base"/>
3175     <Disk ovf:diskId="lamp-app" ovf:fileRef="lamp-app" ovf:capacity="4294967296"
3176         ovf:populatedSize="2349692"
3177
3178 ovf:format="http://www.vmware.com/specifications/vmdk.html#streamOptimized"
3179     ovf:parentRef="lamp-base"/>
3180 </DiskSection>
3181 <!-- Describes all networks used in the package -->
3182 <NetworkSection>
3183     <Info>Logical networks used in the package</Info>
3184     <Network ovf:name="VM Network">
3185         <Description>The network that the LAMP Service is available
3186             on</Description>
3187     </Network>
3188 </NetworkSection>
3189 <VirtualSystemCollection ovf:id="LampService">
3190     <Info>Virtual appliance with a 2-tier distributed LAMP stack</Info>
3191     <Name>LAMP Service</Name>
3192     <!-- Overall information about the product -->
3193     <ProductSection ovf:class="org.mylamp">
3194         <Info>Product information for the service</Info>
3195         <Product>My Lamp Service</Product>
3196         <Version>1.0</Version>
3197         <FullVersion>1.0.0</FullVersion>
3198     </ProductSection>
3199     <ProductSection ovf:class="org.linuxdist">
3200         <Info>Product customization for Operating System Level</Info>
3201         <Product>Linux Distribution X</Product>
3202         <Version>2.6.3</Version>
3203         <Property ovf:key="dbHostname" ovf:type="string">
3204             <Description>Specifies the hostname for database virtual
3205                 machine</Description>
3206         </Property>
3207         <Property ovf:key="appHostname" ovf:type="string">
3208             <Description>Specifies the hostname for application server virtual
3209                 machine</Description>
3210         </Property>
3211         <Property ovf:key="dbIp" ovf:type="string">
3212             <Description>Specifies the IP address for the database virtual
3213                 machine</Description>
3214         </Property>
3215         <Property ovf:key="appIp" ovf:type="string">
3216             <Description>Specifies the IP address for application server
3217                 VM</Description>
3218         </Property>
3219         <Property ovf:key="subnet" ovf:type="string">
3220             <Description> Specifies the subnet to use on the deployed network
3221                 </Description>
3222         </Property>
3223         <Property ovf:key="gateway" ovf:type="string">
3224             <Description> Specifies the gateway on the deployed network
3225                 </Description>
3226         </Property>
3227         <Property ovf:key="dns" ovf:type="string">
3228             <Description> A comma separated list of DNS servers on the deployed
3229                 network </Description>
3230         </Property>
3231         <Property ovf:key="netCoreRmemMaxMB" ovf:type="uint16" ovf:value="16"

```



```

3232         ovf:userConfigurable="true">
3233         <Description> Specify TCP read max buffer size in mega bytes. Default
3234 is
3235         16. </Description>
3236     </Property>
3237     <Property ovf:key="netCoreWmemMaxMB" ovf:type="uint16" ovf:value="16"
3238         ovf:userConfigurable="true">
3239         <Description> Specify TCP write max buffer size in mega bytes. Default
3240 is
3241         16. </Description>
3242     </Property>
3243 </ProductSection>
3244 <!-- Apache component configuration parameters -->
3245 <ProductSection ovf:class="org.apache.httpd">
3246     <Info>Product customization for the installed Apache Web Server</Info>
3247     <Product>Apache Distribution Y</Product>
3248     <Version>2.6.6</Version>
3249     <Property ovf:key="httpPort" ovf:type="uint16" ovf:value="80"
3250         ovf:userConfigurable="true">
3251         <Description>Port number for HTTP requests</Description>
3252     </Property>
3253     <Property ovf:key="httpsPort" ovf:type="uint16" ovf:value="443"
3254         ovf:userConfigurable="true">
3255         <Description>Port number for HTTPS requests</Description>
3256     </Property>
3257     <Property ovf:key="startThreads" ovf:type="uint16" ovf:value="50"
3258         ovf:userConfigurable="true">
3259         <Description>Number of threads created on startup. </Description>
3260     </Property>
3261     <Property ovf:key="minSpareThreads" ovf:type="uint16" ovf:value="15"
3262         ovf:userConfigurable="true">
3263         <Description>Minimum number of idle threads to handle request spikes.
3264         </Description>
3265     </Property>
3266     <Property ovf:key="maxSpareThreads" ovf:type="uint16" ovf:value="30"
3267         ovf:userConfigurable="true">
3268         <Description>Maximum number of idle threads </Description>
3269     </Property>
3270     <Property ovf:key="maxClients" ovf:type="uint16" ovf:value="256"
3271         ovf:userConfigurable="true">
3272         <Description>Limits the number of simultaneous requests that are
3273         served. </Description>
3274     </Property>
3275 </ProductSection>
3276 <!-- MySQL component configuration parameters -->
3277 <ProductSection ovf:class="org.mysql.db">
3278     <Info>Product customization for the installed MySql Database Server</Info>
3279     <Product>MySQL Distribution Z</Product>
3280     <Version>5.0</Version>
3281     <Property ovf:key="queryCacheSizeMB" ovf:type="uint16" ovf:value="32"
3282         ovf:userConfigurable="true">
3283         <Description>Buffer to cache repeated queries for faster access (in
3284         MB)</Description>
3285     </Property>
3286     <Property ovf:key="maxConnections" ovf:type="uint16" ovf:value="500"
3287         ovf:userConfigurable="true">
3288         <Description>The number of concurrent connections that can be
3289         served</Description>
3290     </Property>
3291     <Property ovf:key="waitTimeout" ovf:type="uint16" ovf:value="100"
3292         ovf:userConfigurable="true">
3293         <Description>Number of seconds to wait before timing out a connection
3294         </Description>
3295     </Property>

```

```

3296     </ProductSection>
3297     <!-- PHP component configuration parameters -->
3298     <ProductSection ovf:class="net.php">
3299         <Info>Product customization for the installed PHP component</Info>
3300         <Product>PHP Distribution U</Product>
3301         <Version>5.0</Version>
3302         <Property ovf:key="sessionTimeout" ovf:type="uint16" ovf:value="5"
3303             ovf:userConfigurable="true">
3304             <Description> How many minutes a session has to be idle before it is
3305                 timed out </Description>
3306         </Property>
3307         <Property ovf:key="concurrentSessions" ovf:type="uint16" ovf:value="500"
3308             ovf:userConfigurable="true">
3309             <Description> The number of concurrent sessions that can be served
3310                 </Description>
3311         </Property>
3312         <Property ovf:key="memoryLimit" ovf:type="uint16" ovf:value="32"
3313             ovf:userConfigurable="true">
3314             <Description> How much memory in megabytes a script can consume before
3315                 being killed </Description>
3316         </Property>
3317     </ProductSection>
3318     <StartupSection>
3319         <Info>Startup order of the virtual machines</Info>
3320         <Item ovf:id="DbServer" ovf:order="1" ovf:startDelay="120"
3321             ovf:startAction="powerOn" ovf:waitingForGuest="true"
3322             ovf:stopDelay="120"
3323             ovf:stopAction="guestShutdown"/>
3324         <Item ovf:id="AppServer" ovf:order="2" ovf:startDelay="120"
3325             ovf:startAction="powerOn" ovf:waitingForGuest="true"
3326             ovf:stopDelay="120"
3327             ovf:stopAction="guestShutdown"/>
3328     </StartupSection>
3329     <VirtualSystem ovf:id="AppServer">
3330         <Info>The configuration of the AppServer virtual machine</Info>
3331         <Name>Application Server</Name>
3332         <!-- Linux component configuration parameters -->
3333         <ProductSection ovf:class="org.linuxdistx">
3334             <Info>Product customization for the installed Linux system</Info>
3335             <Product>Linux Distribution X</Product>
3336             <Version>2.6.3</Version>
3337             <Property ovf:key="hostname" ovf:type="string"
3338                 ovf:value="{appHostName}"/>
3339             <Property ovf:key="ip" ovf:type="string" ovf:value="{appIp}"/>
3340             <Property ovf:key="subnet" ovf:type="string" ovf:value="{subnet}"/>
3341             <Property ovf:key="gateway" ovf:type="string" ovf:value="{gateway}"/>
3342             <Property ovf:key="dns" ovf:type="string" ovf:value="{dns}"/>
3343             <Property ovf:key="netCoreRmemMaxMB" ovf:type="string"
3344                 ovf:value="{netCoreRmemMaxMB}"/>
3345             <Property ovf:key="netCoreWmemMaxMB" ovf:type="string"
3346                 ovf:value="{netCoreWmemMaxMB}"/>
3347         </ProductSection>
3348         <OperatingSystemSection ovf:id="99">
3349             <Info>Guest Operating System</Info>
3350             <Description>Linux 2.6.x</Description>
3351         </OperatingSystemSection>
3352         <VirtualHardwareSection>
3353             <Info>Virtual Hardware Requirements: 256 MB, 1 CPU, 1 disk, 1
3354             NIC</Info>
3355             <System>
3356                 <vssd:ElementName>Virtual Hardware Family</vssd:ElementName>
3357                 <vssd:InstanceID>0</vssd:InstanceID>
3358                 <vssd:VirtualSystemType>vmx-04</vssd:VirtualSystemType>
3359             </System>

```

```

3360         <Item>
3361             <rasd:Description>Number of virtual CPUs</rasd:Description>
3362             <rasd:ElementName>1 virtual CPU</rasd:ElementName>
3363             <rasd:InstanceID>1</rasd:InstanceID>
3364             <rasd:ResourceType>3</rasd:ResourceType>
3365             <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
3366         </Item>
3367         <Item>
3368             <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
3369             <rasd:Description>Memory Size</rasd:Description>
3370             <rasd:ElementName>256 MB of memory</rasd:ElementName>
3371             <rasd:InstanceID>2</rasd:InstanceID>
3372             <rasd:ResourceType>4</rasd:ResourceType>
3373             <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
3374         </Item>
3375         <Item>
3376             <rasd:AutomaticAllocation>true</rasd:AutomaticAllocation>
3377             <rasd:Connection>VM Network</rasd:Connection>
3378             <rasd:ElementName>Ethernet adapter on "VM
3379 Network"</rasd:ElementName>
3380             <rasd:InstanceID>3</rasd:InstanceID>
3381             <rasd:ResourceSubType>PCNet32</rasd:ResourceSubType>
3382             <rasd:ResourceType>10</rasd:ResourceType>
3383         </Item>
3384         <Item>
3385             <rasd:ElementName>SCSI Controller 0 - LSI Logic</rasd:ElementName>
3386             <rasd:InstanceID>4</rasd:InstanceID>
3387             <rasd:ResourceSubType>LsiLogic</rasd:ResourceSubType>
3388             <rasd:ResourceType>6</rasd:ResourceType>
3389         </Item>
3390         <Item>
3391             <rasd:ElementName>Harddisk 1</rasd:ElementName>
3392             <rasd:HostResource>ovf:/disk/lamp-app</rasd:HostResource>
3393             <rasd:InstanceID>5</rasd:InstanceID>
3394             <rasd:Parent>4</rasd:Parent>
3395             <rasd:ResourceType>17</rasd:ResourceType>
3396         </Item>
3397     </VirtualHardwareSection>
3398 </VirtualSystem>
3399 <VirtualSystem ovf:id="DB Server">
3400     <Info>The configuration of the database virtual machine</Info>
3401     <Name>Database Server</Name>
3402     <!-- Linux component configuration parameters -->
3403     <ProductSection ovf:class="org.linuxdistx">
3404         <Info>Product customization for the installed Linux system</Info>
3405         <Product>Linux Distribution X</Product>
3406         <Version>2.6.3</Version>
3407         <Property ovf:key="hostname" ovf:type="string"
3408             ovf:value="{dbHostName}"/>
3409         <Property ovf:key="ip" ovf:type="string" ovf:value="{dbIp}"/>
3410         <Property ovf:key="subnet" ovf:type="string" ovf:value="{subnet}"/>
3411         <Property ovf:key="gateway" ovf:type="string" ovf:value="{gateway}"/>
3412         <Property ovf:key="dns" ovf:type="string" ovf:value="{dns}"/>
3413         <Property ovf:key="netCoreRmemMaxMB" ovf:type="string"
3414             ovf:value="{netCoreRmemMaxMB}"/>
3415         <Property ovf:key="netCoreWmemMaxMB" ovf:type="string"
3416             ovf:value="{netCoreWmemMaxMB}"/>
3417     </ProductSection>
3418     <OperatingSystemSection ovf:id="99">
3419         <Info>Guest Operating System</Info>
3420         <Description>Linux 2.6.x</Description>
3421     </OperatingSystemSection>
3422 </VirtualHardwareSection>
3423     <Info>Virtual Hardware Requirements: 256 MB, 1 CPU, 1 disk, 1

```

```

3424 nic</Info>
3425     <System>
3426         <vssd:ElementName>Virtual Hardware Family</vssd:ElementName>
3427         <vssd:InstanceID>0</vssd:InstanceID>
3428         <vssd:VirtualSystemType>vmx-04</vssd:VirtualSystemType>
3429     </System>
3430     <Item>
3431         <rasd:Description>Number of virtual CPUs</rasd:Description>
3432         <rasd:ElementName>1 virtual CPU</rasd:ElementName>
3433         <rasd:InstanceID>1</rasd:InstanceID>
3434         <rasd:ResourceType>3</rasd:ResourceType>
3435         <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
3436     </Item>
3437     <Item>
3438         <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
3439         <rasd:Description>Memory Size</rasd:Description>
3440         <rasd:ElementName>256 MB of memory</rasd:ElementName>
3441         <rasd:InstanceID>2</rasd:InstanceID>
3442         <rasd:ResourceType>4</rasd:ResourceType>
3443         <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
3444     </Item>
3445     <Item>
3446         <rasd:AutomaticAllocation>true</rasd:AutomaticAllocation>
3447         <rasd:Connection>VM Network</rasd:Connection>
3448         <rasd:ElementName>Ethernet adapter on "VM
3449 Network"</rasd:ElementName>
3450         <rasd:InstanceID>3</rasd:InstanceID>
3451         <rasd:ResourceType>10</rasd:ResourceType>
3452     </Item>
3453     <Item>
3454         <rasd:ElementName>SCSI Controller 0 - LSI Logic</rasd:ElementName>
3455         <rasd:InstanceID>4</rasd:InstanceID>
3456         <rasd:ResourceSubType>LsiLogic</rasd:ResourceSubType>
3457         <rasd:ResourceType>6</rasd:ResourceType>
3458     </Item>
3459     <Item>
3460         <rasd:ElementName>Harddisk 1</rasd:ElementName>
3461         <rasd:HostResource>ovf:/disk/lamp-db</rasd:HostResource>
3462         <rasd:InstanceID>5</rasd:InstanceID>
3463         <rasd:Parent>4</rasd:Parent>
3464         <rasd:ResourceType>17</rasd:ResourceType>
3465     </Item>
3466     </VirtualHardwareSection>
3467 </VirtualSystem>
3468 </VirtualSystemCollection>
3469 </Envelope>

```

3470

3471
3472
3473

ANNEX E (informative) Extensibility example

3474 The OVF specification allows custom metadata to be added to OVF descriptors in several ways:

- 3475 • New section elements can be defined as part of the `Section` substitution group, and used
3476 wherever the OVF Schemas allow sections to be present.
- 3477 • The OVF Schemas use an open content model, where all existing types can be extended at the
3478 end with additional elements. Extension points are declared in the OVF Schemas with `xs:any`
3479 declarations with `namespace="##other"`.
- 3480 • The OVF Schemas allow additional attributes on existing types.

3481 Custom metadata is not allowed to use OVF XML namespaces. On custom elements, a Boolean
3482 `ovf:required` attribute specifies whether the information in the element is required for correct behavior
3483 or optional.

3484 The open content model in the OVF Schemas only allows extending existing types at the end. Using XML
3485 Schema 1.0, it is not easy to allow for a more flexible open content model, due to the Unique Particle
3486 Attribution rule and the necessity of adding `xs:any` declarations everywhere in the schema. The XML
3487 Schema 1.1 draft standard contains a much more flexible open content mechanism, using
3488 `xs:openContent mode="interleave"` declarations.

3489 E.1 Custom schema

3490 A custom XML schema defining two extension types is listed below. The first declaration defines a
3491 custom member of the OVF `Section` substitution group, while the second declaration defines a simple
3492 custom type.

```

3493 <?xml version="1.0" encoding="UTF-8"?>
3494 <xs:schema
3495   targetNamespace="http://schemas.customextension.org/1"
3496   xmlns:custom="http://schemas.customextension.org/1"
3497   xmlns="http://schemas.customextension.org/1"
3498   xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1"
3499   xmlns:xs="http://www.w3.org/2001/XMLSchema"
3500   attributeFormDefault="qualified"
3501   elementFormDefault="qualified">
3502
3503   <!-- Define a custom member of the ovf:Section substitution group -->
3504   <xs:element name="CustomSection" type="custom:CustomSection_Type"
3505 substitutionGroup="ovf:Section"/>
3506
3507   <xs:complexType name="CustomSection_Type">
3508     <xs:complexContent>
3509       <xs:extension base="ovf:Section_Type">
3510         <xs:sequence>
3511           <xs:element name="Data" type="xs:string"/>
3512         </xs:sequence>
3513         <xs:anyAttribute namespace="##any" processContents="lax"/>
3514       </xs:extension>
3515     </xs:complexContent>
3516   </xs:complexType>
3517
3518   <!-- Define other simple custom type not part of ovf:Section substitution group --
3519 >
3520   <xs:complexType name="CustomOther_Type">
3521
```

```

3522     <xs:sequence>
3523         <xs:element name="Data" type="xs:string"/>
3524     </xs:sequence>
3525     <xs:attribute ref="ovf:required"/>
3526     <xs:anyAttribute namespace="##any" processContents="lax"/>
3527 </xs:complexType>
3528
3529 </xs:schema >

```

3530 E.2 Descriptor with custom extensions

3531 A complete OVF descriptor using the custom schema above is listed below. The descriptor validates
 3532 against the OVF Schema and the custom schema, but apart from extension examples, the descriptor is
 3533 kept minimal.

3534 The descriptor contains all three extension types: a custom OVF `Section` element, a custom element at
 3535 an extension point, and a custom attribute.

```

3536 <?xml version="1.0" encoding="UTF-8"?>
3537 <Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3538     xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-
3539 schema/2/CIM_VirtualSystemSettingData"
3540     xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
3541 schema/2/CIM_ResourceAllocationSettingData"
3542     xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1"
3543     xmlns="http://schemas.dmtf.org/ovf/envelope/1"
3544     xmlns:custom="http://schemas.customextension.org/1">
3545
3546     <!-- Dummy References element -->
3547     <References/>
3548
3549     <!-- EXAMPLE: Optional custom OVF section element with validation against custom
3550 schema -->
3551     <custom:CustomSection ovf:required="false">
3552         <Info>Description of custom extension</Info>
3553         <custom:Data>somevalue</custom:Data>
3554     </custom:CustomSection>
3555
3556     <!-- Describes all networks used in the package -->
3557     <NetworkSection>
3558         <Info>Logical networks used in the package</Info>
3559         <!-- EXAMPLE: Optional custom attribute -->
3560         <Network ovf:name="VM Network" custom:desiredCapacity="1 Gbit/s"/>
3561         <!-- EXAMPLE: Optional custom metadata inserted at extension point with
3562 validation against custom schema -->
3563         <custom:CustomOther xsi:type="custom:CustomOther_Type" ovf:required="false">
3564             <custom:Data>somevalue</custom:Data>
3565         </custom:CustomOther>
3566     </NetworkSection>
3567
3568     <!-- Dummy Content element -->
3569     <VirtualSystem ovf:id="Dummy">
3570         <Info>Dummy VirtualSystem</Info>
3571     </VirtualSystem>
3572 </Envelope>

```

3573 The OVF environment XML Schemas contain extension mechanisms matching those of the OVF
 3574 envelope XML Schemas, so OVF environment documents are similarly extensible.

**ANNEX F
(informative)
Change log**

3575
3576
3577

3578

Version	Date	Description
1.0.0	2009-02-17	
1.0.1	2011-10-20	Release for DMTF errata publication
2.0.0	2014-04-24	Release for DMTF publication as Informational

3579