



Document Number: DSP2017

Date: 2013-07-03

Version: 2.0.0a

Open Virtualization Format White Paper

Information for Work-in-Progress version:

This document is subject to change at any time without further notice.

It expires on: 2013-12-31

Provide any comments through the DMTF Feedback Portal:

<http://www.dmtf.org/standards/feedback>

Document Type: DMTF Informational

Document Status: Work in Progress

Document Language: en-US

Copyright notice

Copyright © 2007,2013 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patent owner or claimant, and shall have no liability or responsibility for costs or losses incurred if a standard is withdrawn or modified after publication, and shall be indemnified and held harmless by any party implementing the standard from any and all claims of infringement by a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 comprise 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

Table of Contents

2	1	Introduction.....	7
3	1.1	Overview.....	7
4	1.2	Design Considerations.....	7
5	2	OVF Key Concepts.....	9
6	2.1	Virtual Appliances.....	9
7	2.2	Life-Cycle.....	10
8	3	OVF Package.....	12
9	3.1	General XML concepts used in OVF.....	13
10	3.1.1	Element.....	13
11	3.1.2	Attribute.....	13
12	3.1.3	Substitution Group.....	13
13	3.1.4	Global attributes used in the OVF.....	13
14	3.1.5	Elements used in OVF.....	13
15	3.1.6	Extensibility in OVF.....	15
16	3.2	OVF Top Level Elements.....	16
17	3.2.1	References element.....	16
18	3.2.2	NetworkSection element.....	16
19	3.2.3	DiskSection Element.....	17
20	3.2.4	DeploymentOptionsSection Element.....	17
21	3.3	OVF Section Elements used in Virtual System & Virtual System Collections.....	18
22	3.3.1	AnnotationSection element.....	18
23	3.3.2	ProductSection element.....	18
24	3.3.3	OperatingSystemSection element.....	19
25	3.3.4	EulaSection element.....	19
26	3.3.5	VirtualHardwareSection element.....	20
27	3.3.6	ResourceAllocationSection element.....	22
28	3.3.7	InstallSection element.....	22
29	3.3.8	StartupSection element.....	23
30	3.3.9	EnvironmentFilesSection element.....	25
31	3.3.10	BootDeviceSection element.....	25
32	3.3.11	SharedDiskSection element.....	26
33	3.3.12	ScaleOutSection element.....	28
34	3.3.13	PlacementGroupSection element.....	30
35	3.3.14	PlacementSection element.....	30
36	3.3.15	EncryptionSection element.....	33
37	4	Authoring an OVF package.....	36
38	4.1	Creation.....	36
39	4.2	Internationalization.....	37
40	4.3	Extensibility.....	38
41	4.3.1	Substitution Group.....	39
42	4.3.2	Elements.....	40
43	4.3.3	Attributes.....	41
44	4.4	Conformance.....	41
45	4.5	Virtual Hardware Description.....	41
46	4.6	Example Descriptors.....	43
47	5	Deploying an OVF package.....	43
48	5.1	Deployment.....	43
49	5.2	OVF Environment Descriptor.....	44
50	5.3	Resource Configuration Options During Deployment.....	44
51	5.4	Product Customization During Deployment Using Property Elements.....	45
52	6	Portability.....	46
53		ANNEX A (informative) Single Virtual System Example.....	48

54 ANNEX B (informative) Multi-tiered Pet Store Example 51

55 B.1 Architecture and Packaging 51

56 B.2 Properties 52

57 B.3 Disk Layout 53

58 B.4 Pet Store OVF Descriptor 53

59 B.5 Complete OVF Environment 60

60 ANNEX C (informative) Single Virtual System LAMP Stack Example 62

61 C.1 Deployment-time Customization 62

62 C.2 Simple LAMP OVF Descriptor 64

63 ANNEX D (informative) Multiple Virtual System LAMP Stack Example 68

64 D.1 Two-tier LAMP OVF Descriptor 68

65 ANNEX E (informative) Extensibility Example 74

66 E.1 Custom Schema 74

67 E.2 Descriptor with custom extensions 75

68 ANNEX F (informative) Change Log 76

69 **Figures**

70 Figure 1 – OVF Package Lifecycle 10

71 Figure 2 – OVF Author Function 11

72 Figure 3 – OVF Deployment Function 12

73 Figure 4 - StartupSection Traversal 24

74 Figure 5 – Pet Store OVF Package 51

75 Figure 6 – Pet Store Virtual Disk Layout 53

76 **Tables**

77 Table 1 – Web Tier Configuration 52

78 Table 2 – Database Tier Configuration 52

79 Table 3 – LAMP Configuration 62

80

81

82

Foreword

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

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

87 Lawrence Lamers	VMware Inc. (Chair)
88 Marvin Waschke	DMTF Fellow (co-Editor)
89 Peter Wörndle	Ericsson AB (co-Editor)
90 Eric Wells	Hitachi, Ltd.(co-Editor)
91	
92 Hemal Shah	Broadcom Corporation
93 Shishir Pardikar	Citrix Systems Inc.
94 Richard Landau	DMTF Fellow
95 Robert Freund	Hitachi, Ltd.
96 Jeff Wheeler	Huawei
97 Monica Martin	Microsoft Corporation
98 Cheng Wei	Microsoft Corporation
99 Srinivas Maturi	Oracle
100 Steffen Grarup	VMware Inc
101 Rene Schmidt	VMware Inc.
102 Ghazanfar Ali	ZTE Corporation

103

104 **1 Introduction**

105 **1.1 Overview**

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

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

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

- 114 1. The ability for ISV's to package a software solution that is capable of being used on more than
115 one hypervisor.
- 116 2. The ability to package a virtual system or collection of virtual systems so they can be moved from
117 one data center to another.

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

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

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

122 OVF 2 adds the following features:

- 123 • Support for Network Ports
- 124 • Scaling at deployment time
- 125 • Support for basic placement policies
- 126 • Encryption of OVF packages
- 127 • Disk sharing at runtime
- 128 • Advanced Device Boot Order
- 129 • Advanced Data Transfer to Guest OS
- 130 • Support for Improved Internationalization - I18N
- 131 • Support of HASH Improved
- 132 • Updated CIM schema

133 OVF has adopted the Common Information Model (CIM), where appropriate, to allow management
134 software to clearly understand and easily map resource properties using an open standard. The
135 CIM_ResourceAllocationSettingData class and its sub-classes for specific device types is used to specify
136 the resources needed for the virtual system to operate.

137 OVF 2 supports network configuration and the IEEE Edge Virtual Bridging Discovery and Configuration
138 protocols. This feature uses DSP8049 Network Port Profile to accomplish this. The
139 CIM_EthernetPortAllocationSettingData class provides the essential properties.

140 This document aims to give details of the motivation, goals, design and expected usage of OVF, and
141 should be read in accompaniment to the OVF specification of the same revision number.

142 **1.2 Design Considerations**

143 The rapid adoption of virtual infrastructure has highlighted the need for a standard, portable meta-data
144 format for the distribution of virtual systems onto and between virtualization platforms. The ability to

145 package a software application together with the operating system on which it is certified, into a format
146 that can be easily transferred from an ISV, through test and development and into production as a pre-
147 configured, pre-packaged unit with no external dependencies, is extremely attractive. Such pre-deployed,
148 ready to run applications packaged with the configuration of the virtual systems required to run them are
149 called virtual appliances. In order to make this concept practical on a broad scale it is important that the
150 industry adopts a vendor-neutral standard for packaging such virtual appliances and the meta-data
151 required to automatically and securely install, configure, and run them on any virtualization platform.

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

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

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

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

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

176 The OVF specification is complimentary to existing IT management standards and frameworks and
177 promotes best-of-breed competition through openness and extensibility. The explicit copyright notice
178 attached to this document is intended to avoid arbitrary, independent, piece-wise extensions to the format
179 while permitting free distribution and implementation of the specification.

180

181

182 2 OVF Key Concepts

183 2.1 Virtual Appliances

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

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

201 Virtual appliances need not be developed and delivered by 3rd party ISVs – the concept is equally useful
202 and often used within an enterprise in which a virtual system template for a particular service is
203 assembled, tested, and certified by an IT organization and then packaged for repeated, “cookie cutter”
204 deployment throughout the enterprise.

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

211 For example, consider a typical web application that consists of three tiers. A web tier that implements the
212 presentation logic, and application server tier that implements the business logic, and a back-end
213 database tier. A straightforward implementation would divide this into 3 virtual systems, one for each tier.
214 In this way, the application can scale from a fraction of a single physical host to 3 physical hosts. Another
215 approach is to treat each tier as a service in itself. Hence, each tier can scale to a multi-virtual system
216 service that provides a clustered solution. Again, taking the web-application as an example, a common
217 scenario is to have many web servers, fewer applications servers, and one or two database servers.
218 Implemented as virtual systems, each tier can scale across as many or as few physical machines as
219 required, and each tier can support multiple instances of virtual systems that service requests.

220 Transport virtual system templates. One OVF may contain a single or many virtual systems (it is left to the
221 developer to decide which arrangement best suits their application). OVFs must be installed before they
222 can be run; a particular virtualization platform may run the virtual system from the OVF, but this is not
223 required. If this is done, the OVF itself can no longer be viewed as a “golden image” version of the
224 appliance, since run-time state for the virtual system(s) will pervade the OVF. Moreover the digital
225 signature that allows the platform to check the integrity of the OVF will be invalid.

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

230 system with multiple disks, or multiple virtual systems, or need customization of the virtual system at
 231 install time, or if the virtual system is intended to run on multiple virtualization platforms (even if the
 232 virtualization platforms claim support of the particular virtual hard disk format used).

233 Included within the OVF remit is the concept of the certification and integrity of a packaged virtual
 234 appliance, allowing the platform to determine the provenance of the appliance and to allow the end-user
 235 to make the appropriate trust decisions. The OVF specification has been constructed so that the
 236 appliance is responsible for its own configuration and modification. In particular, this means that the
 237 virtualization platform does not need to be able to read from the appliance's file systems. This decoupling
 238 of platform from the appliance means that OVF packages may be implemented using any operating
 239 system, and installed on any virtualization platform that supports the OVF format. A specific mechanism is
 240 provided for appliances to detect and react to the platform on which they are installed. This allows
 241 platforms to extend this specification in unique ways without breaking compatibility of appliances across
 242 the industry.

243 The OVF format has several specific features that are designed for complex, multi-tier services and their
 244 associated distribution, installation, configuration and execution workflow:

245 It directly supports the configuration of multi-tier applications and the composition of virtual systems to
 246 deliver composed services.

247 It permits the specification of both virtual system and application-level configuration.

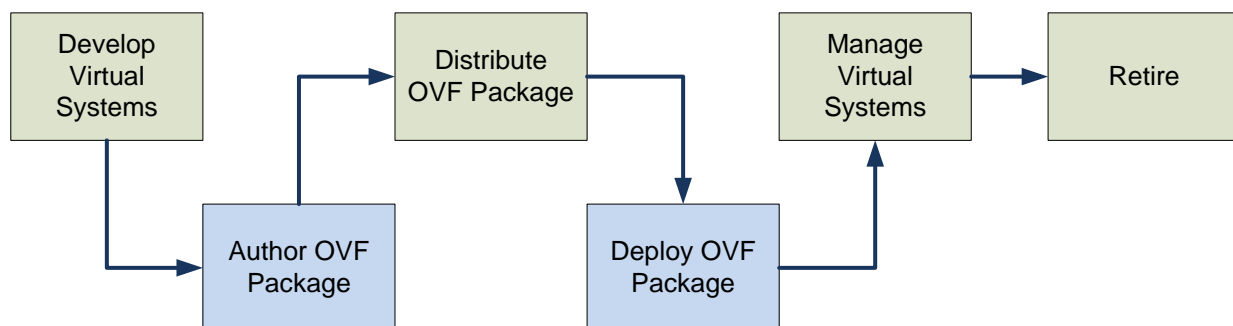
248 It offers robust mechanisms for validation of the contents of the OVF, and full support for unattended
 249 installation to ease the burden of deployment for users, and thereby enhance the user's experience.

250 It uses commercially accepted procedures for integrity checking of the contents of the OVF, through the
 251 use of signatures and trusted third parties. This serves to reassure the consumer of an appliance that it
 252 has not been modified since signed by the creator of the appliance. This is seen as critical to the success
 253 of the virtual appliance market, and to the viability of independent creation and online download of
 254 appliances.

255 It permits commercial interests of the appliance vendor and user to be respected, by providing a basic
 256 method for presentation and acknowledgement of licensing terms associated with the appliance.

257 2.2 Life-Cycle

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



259

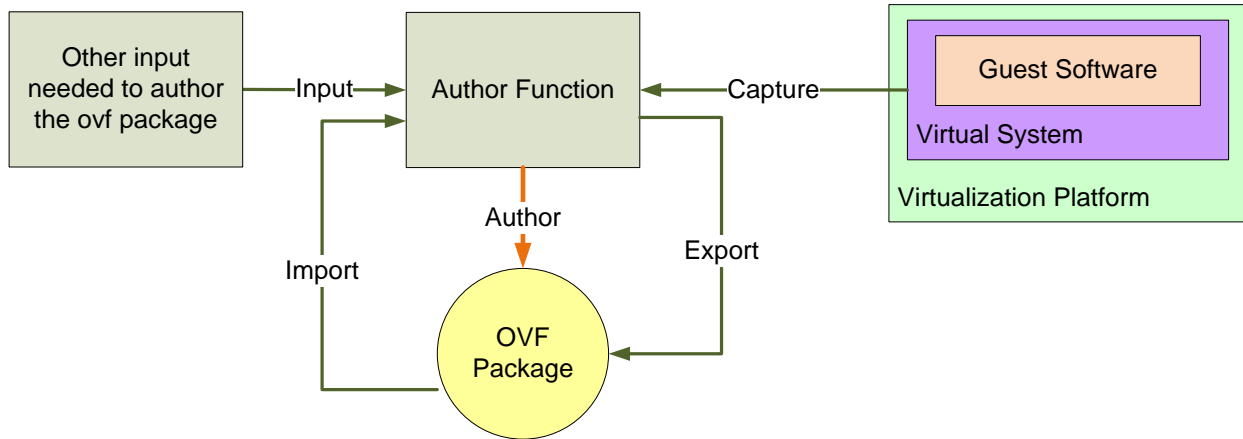
260

Figure 1 – OVF Package Lifecycle

261 An OVF package is built from components that have been developed or acquired by the OVF author.
 262 These are packaged into a set of files that comprise a virtual appliance, consisting of one or more virtual
 263 machines and virtual machine collections and the relevant configuration and deployment meta data. For
 264 example, a clustered database component might be acquired from a third-party ISV. The installed service
 265 is then managed and eventually retired. Distribution, management and retirement are outside the scope
 266 of OVF and are specific to the virtualization product used and the virtual appliance installed from the OVF.

267 Management includes, for example, ongoing maintenance, configuration, and upgrade of the appliance.
 268 These activities depend on the installed service and environment, not the OVF package. The OVF
 269 specification deals specifically with the authoring and deployment phases.

270 The OVF author function is illustrated in **Figure 2**.



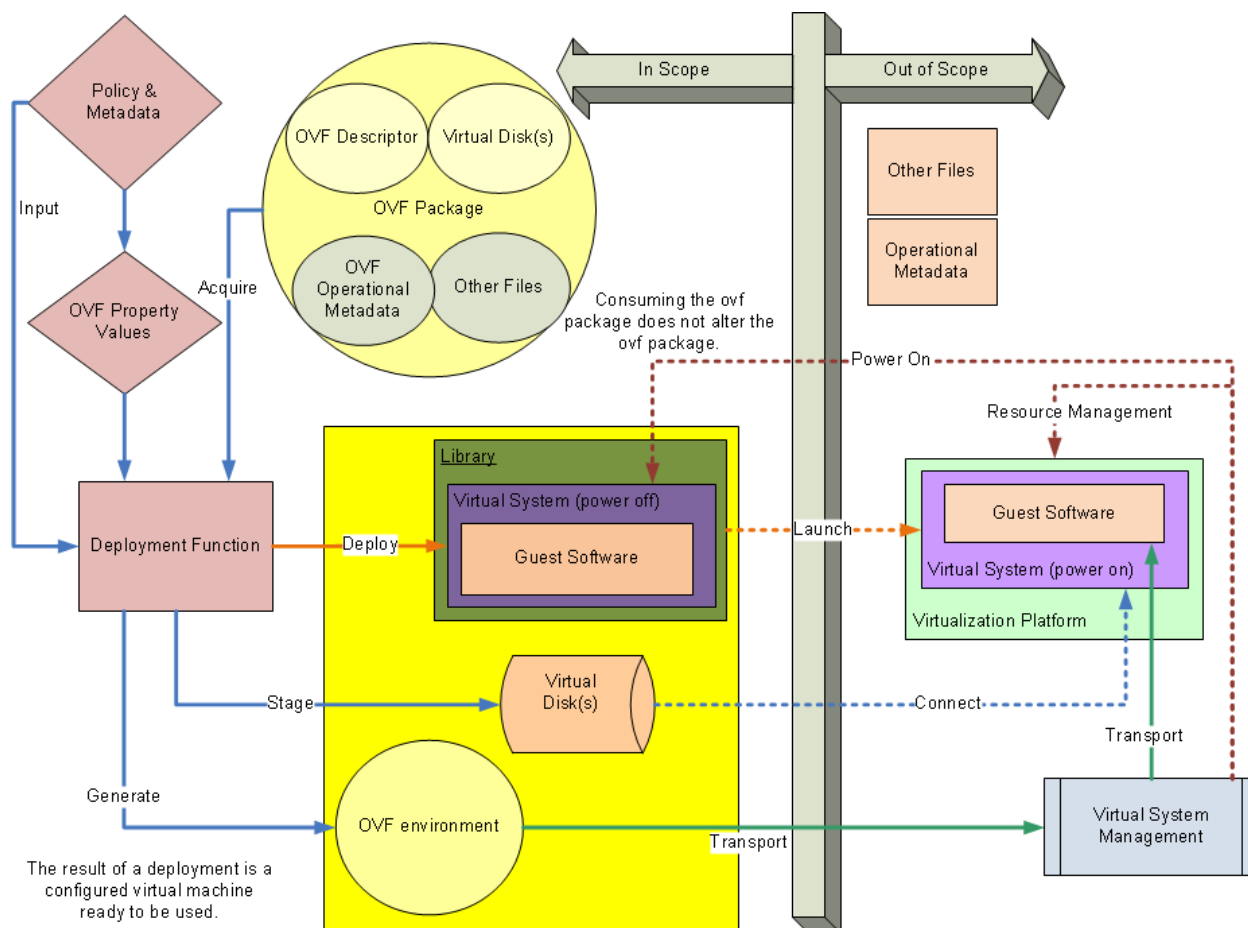
271

272

Figure 2 – OVF Author Function

273 In practice, OVFs are usually authored in two ways. The straight forward method is to use a text editor or
 274 an XML authoring tool to write an OVF XML descriptor following the OVF specification, then assemble the
 275 required images, and other files. An alternative method is also often used. This method relies on the
 276 ability of some virtualization platforms to export an OVF for a collection of virtual systems or machines. An
 277 OVF is exported from a running appliance. The “raw” exported OVF may be edited to become a new OVF
 278 package. This may be done for various reasons: improving portability between virtualization platforms or
 279 providing options for configuration are two common reasons. This edited exported OVF becomes a new
 280 OVF package.

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



283

284

Figure 3 – OVF Deployment Function

285 The OVF operational metadata is information that may be needed for the proper operation of the virtual
 286 system or collection of virtual systems. The OVF operational metadata is a sub-set of the operational
 287 metadata that may be available when the virtual system(s) is powered on.

288 As shown in the diagram, the OVF deployment function transports the OVF environment to the
 289 virtualization platform. The OVF specification is flexible on the exact nature of the transport, but it can be
 290 thought of as placing media, like a CD ROM, into a virtual reader on the virtual machine and the media
 291 being read by the guest operating system each time the system starts up. The meta-data in the OVF
 292 environment is used for configuration after operating system startup and meeting other requirements of
 293 guest software and virtualization platform for the proper operation of the virtual appliance.

294 3 OVF Package

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

299 **3.1 General XML concepts used in OVF**

300 **3.1.1 Element**

301 An XML element is data container in the OVF descriptor. Each element begins and ends with an element
302 type, i.e., element name, contained in an element tag. The element's start tag, e.g., <elementname> and
303 the element's end tag, e.g., </elementname> delineate the element contents. An element can contain:
304 other elements, text, attributes, or a mix of all of the these.

305 XML elements follow these naming rules:

- 306 • Names can contain letters, numbers, and other characters
- 307 • Names cannot start with a number or punctuation character
- 308 • Names cannot start with the letters xml (or XML, or Xml, etc)
- 309 • Names cannot contain spaces
- 310 • Any name can be used, no words are reserved.

311 **3.1.2 Attribute**

312 XML elements can have attributes that provide additional information about an element. Attributes often
313 provide information that is not a part of the data. Attribute values are always be quoted using single or
314 double quotes.

315 **3.1.3 Substitution Group**

316 A substitution group is a feature of XML schema that allows you to specify elements that can replace
317 another element in documents generated from that schema. The replaceable element is called the head
318 element and is defined in the schema's global scope. The elements of the substitution group are of the
319 same type as the head element or a type that is derived from the head element's type.

320 In essence, a substitution group allows you to build a collection of elements that can be specified using a
321 generic element. For example, if you are building an ordering system for a company that sells three types
322 of widgets you might define a generic widget element that contains a set of common data for all three
323 widget types. Then you can define a substitution group that contains a more specific set of data for each
324 type of widget. In your contract you can then specify the generic widget element as a message part
325 instead of defining a specific ordering operation for each type of widget. When the actual message is
326 built, the message can contain any of the elements of the substitution group.

327 **3.1.4 Global attributes used in the OVF**

328 The following global attributes are defined in the OVF schema. Element specific attributes are also
329 defined see 3.2.and 3.3

- 330 • required xs:boolean default="true" - determines whether import fails if the Section is not
331 understood
- 332 • transport xs:string default="" - space-separated list of supported transport
333 types</xs:documentation
- 334 • configuration xs:string - configuration from the DeploymentOptionSection element that this entry
335 is valid for
- 336 • bound xs:string - range marker entry

337 **3.1.5 Elements used in OVF**

338 The root element of an OVF Descriptor is the Envelope element

339 The OVF schema uses substitution groups to provide extensibility. Two substitution groups are defined:

- 340 • Section element - a substitution group that contains section elements
- 341 • Content element – a substitution group that contains a virtual system element(s) or a collection
- 342 of virtual systems element(s)

343 Elements that are direct children of an Envelope element:

- 344 • References element - a sequence of references to all external files
- 345 • NetworkSection element
- 346 • DiskSection element
- 347 • DeploymentOptionSection element
- 348 • Content element
- 349 • Strings element

350 Elements that are a substitution group for a Content element:

- 351 • VirtualSystem element
- 352 • VirtualSystemCollection element (may be nested)

353 Elements that are a substitution group for a Section element used in a Content element:

- 354 • AnnotationSection element
- 355 • ProductSection element
- 356 • OperatingSystemSection element
- 357 • EulaSection element
- 358 • VirtualHardwareSection element
- 359 • ResourceAllocationSection element
- 360 • InstallSection element
- 361 • StartupSection element
- 362 • EnvironmentFilesSection element
- 363 • BootDeviceSection element
- 364 • SharedDiskSection element
- 365 • ScaleOutSection element
- 366 • PlacementGroupSection element
- 367 • PlacementSection element
- 368 • EncryptionSection element

369 These elements have to be in the above order when used in an OVF descriptor.

370 Basic structure of an OVF Package:

```
371 ovf:Envelope
372 <xs:element name="References" type="ovf:References_Type">
373
374 <xs:element ref="ovf:Section" minOccurs="0" maxOccurs="unbounded">
375 <DiskSection>
376   <Info> Describes all networks used in the package </Info>
377 <NetworkSection>
```

```

378     <Info>List of logical networks used in the package</Info>
379 <DeploymentOptionSection>
380     <Info>List of deployment options available in the package</Info>
381
382 <xs:element ref="ovf:Content">
383     <VirtualSystemCollection ovf:id="Acme VSC">
384         <Info>The packaging of the first virtual appliance</Info>
385
386         <VirtualSystem ovf:id="Acme VS 1">
387             <Info>The packaging of the virtual machine 1</Info>
388         <VirtualSystem ovf:id="Acme VS 2">
389             <Info>The packaging of the virtual machine 2</Info>
390
391         <VirtualSystemCollection ovf:id="Widget VSC">
392             <Info>The packaging of the second virtual appliance</Info>
393
394             <VirtualSystem ovf:id="Acme VS 3">
395                 <Info>The packaging of the virtual machine 3</Info>
396             <VirtualSystem ovf:id="Acme VS 3">
397                 <Info>The packaging of the virtual machine 4</Info>
398
399 <xs:element name="Strings" type="ovf:Strings_Type" minOccurs="0"
400 maxOccurs="unbounded">
401     <Info> Root element of I18N string bundle</Info>

```

402 3.1.6 Extensibility in OVF

403 The OVF schema use the XSD elements 'any' and 'anyAttribute' to extend the OVF descriptor and
 404 the OVF Environment to provide custom metadata. This feature allows OVF packages to meet a wide
 405 variety of use cases in the industry.

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

- 408 • any - Enables the author to extend the XML document with elements not specified by the
 409 schema
- 410 • anyAttribute - Enables the author to extend the XML document with attributes not specified
 411 by the schema
- 412 • ##any - elements from any namespace are allowed (this is default)
- 413 • ##other - elements from any namespace that is not the namespace of the parent element can
 414 be present

415 An extension at Envelope element level is done by defining a new member of the ovf:Section substitution
 416 group. An extension at Content element level is done by defining a new member of the ovf:Section
 417 substitution group. These new section elements can be used where sections are allowed to be present
 418 by the OVF schema. The Info element in each new section element can be used to give meaningful
 419 warnings to users when a new section element is being skipped because the deployment platform does
 420 not understand it.

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

423 Additional attributes are allowed in the OVF schema. Extension points are declared with an
424 `xs:anyAttribute`

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

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

429 EXAMPLE 1:

```
430 <!-- Optional custom section example -->
431 <otherns:IncidentTrackingSection ovf:required="false">
432   <Info>Specifies information useful for incident tracking purposes</Info>
433   <BuildSystem>Acme Corporation Official Build System</BuildSystem>
434   <BuildNumber>102876</BuildNumber>
435   <BuildDate>10-10-2008</BuildDate>
436 </otherns:IncidentTrackingSection>
```

437 EXAMPLE 2:

```
438 <!-- Open content example (extension of existing type) -->
439 <AnnotationSection>
440   <Info>Specifies an annotation for this virtual machine</Info>
441   <Annotation>This is an example of how a future element (Author) can still be
442     parsed by older clients</Annotation>
443   <!-- AnnotationSection extended with Author element -->
444   <otherns:Author ovf:required="false">John Smith</otherns:Author>
445 </AnnotationSection>
```

446 EXAMPLE 3:

```
447 <!-- Optional custom attribute example -->
448 <Network ovf:name="VM network" otherns:desiredCapacity="1 Gbit/s">
449   <Description>The main network for VMs</Description>
450 </Network>
```

451

452 3.2 OVF Top Level Elements

453 3.2.1 References element

454 The references element contains the references to all external files

455 3.2.2 NetworkSection element

456 See clause X for additional information on the `NetworkSection` element

```
457 <NetworkSection>
458   <Info>List of logical networks used in the package</Info>
459   <Network ovf:name="VM Network">
460     <Description>The network that the service will be available on</Description>
461     <NetworkPortProfile>
462       <Item>
463         <epasd:AllocationUnits>GigaBits per Second</epasd:AllocationUnits>
464         <epasd:ElementName>Network Port Profile /epasd:ElementName>
465         <epasd:InstanceID>1</epasd:InstanceID>
466         <epasd:NetworkPortProfileID>1</epasd:NetworkPortProfileID>
```



```

467     <epasd:NetworkPortProfileIDType>4</epasd:NetworkPortProfileIDType>
468     <epasd:Reservation>1</epasd:Reservation>
469   </Item>
470 </NetworkPortProfile>
471 </Network>
472 </NetworkSection>

```

473 3.2.3 DiskSection Element

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

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

478 The virtual disk format may be a simple basic disk block format agnostic to the guest software installed.
 479 For example, VMware VMDK formats deal with 512 byte disk sectors stored in 64KB blocks, in a number
 480 of flat, sparse, and compressed variants. At deployment time, the virtualization platform creates virtual
 481 disks in a basic disk block format it prefers. The runtime virtual disk format may be identical to the
 482 distribution format, but is often different since it may not be efficient to run out of a compressed virtual
 483 disk format. The guest software installed has its own file system format, e.g., NTFS, EXT3, or ZFS. The
 484 OVF virtual disk though does not need to know the file system format.

485 The following example shows a description of virtual disks:

```

486 <DiskSection>
487   <Info>Describes the set of virtual disks</Info>
488   <Disk ovf:diskId="vmdisk1" ovf:fileRef="file1" ovf:capacity="8589934592"
489     ovf:populatedSize="3549324972"
490     ovf:format=
491       "http://www.vmware.com/interfaces/specifications/vmdk.html#sparse">
492   </Disk>
493   <Disk ovf:diskId="vmdisk2" ovf:capacity="536870912">
494   </Disk>
495   <Disk ovf:diskId="vmdisk3" ovf:capacity="{disk.size}"
496     ovf:capacityAllocationUnits="byte * 2^30">
497   </Disk>
498 </DiskSection>

```

499 3.2.4 DeploymentOptionsSection Element

500 The following example illustrates a `DeploymentOptionsSection` element.

```

501 <DeploymentOptionSection>
502   <Configuration ovf:id="minimal">
503     <Label>Minimal</Label>
504     <Description>Some description</Description>
505   </Configuration>
506   <Configuration ovf:id="normal" ovf:default="true">
507     <Label>Typical</Label>
508     <Description>Some description</Description>
509   </Configuration>
510   <!-- Additional configurations -->
511 </DeploymentOptionSection>

```

512 3.3 OVF Section Elements used in Virtual System & Virtual System Collections

513 The following OVF descriptor section elements may appear within a `VirtualSystem` or
514 `VirtualSystemCollection` element.

515 3.3.1 AnnotationSection element

516 The `AnnotationSection` element is user-defined and can appear in `VirtualSystem` and
517 `VirtualSystemCollection` elements. An `AnnotationSection` element is required to contain one and
518 only `Annotation` element. `Annotation` elements are localizable. A suggested use for `Annotation`
519 elements is to display them to the consumers as the package is deployed. Note that the `Annotation`
520 element specified in OVF is not an XML Schema `Annotation` element.

```
521 <AnnotationSection>
522   <Info>An annotation on this service. It can be ignored</Info>
523   <Annotation>Contact customer support if you have any problems</Annotation>
524 </AnnotationSection >
```

525 3.3.2 ProductSection element

526 The `ProductSection` element provides product information such as name and vendor of the
527 appliance and a set of properties that can be used to customize the appliance. These properties will be
528 configured at installation time of the appliance, typically by prompting the user. This is discussed in more
529 detail below.

```
530 <ProductSection ovf:class="com.mycrm.myservice" ovf:instance="1">
531   <Info>Describes product information for the service</Info>
532   <Product>MyCRM Enterprise</Product>
533   <Vendor>MyCRM Corporation</Vendor>
534   <Version>4.5</Version>
535   <FullVersion>4.5-b4523</FullVersion>
536   <ProductUrl>http://www.mycrm.com/enterprise</ProductUrl>
537   <VendorUrl>http://www.mycrm.com</VendorUrl>
538   <Icon ovf:height="32" ovf:width="32" ovf:mimeType="image/png"
539   ovf:fileRef="icon">
540     <Category>Email properties</Category>
541     <Property ovf:key="admin.email" ovf:type="string"
542     ovf:userConfigurable="true">
543       <Label>Admin email</Label>
544       <Description>Email address of administrator</Description>
545     </Property>
546     <Category>Admin properties</Category>
547     <Property ovf:key="app_log" ovf:type="string" ovf:value="low"
548     ovf:userConfigurable="true">
549       <Description>Loglevel for the service</Description>
550     </Property>
551     <Property ovf:key="app_isSecondary" ovf:value="false" ovf:type="boolean">
552       <Description>Cluster setup for application server</Description>
553     </Property>
554     <Property ovf:key="app_ip" ovf:type="string" ovf:value="{appserver-vm}">
555       <Description>IP address of the application server VM</Description>
556     </Property>
```

557 `</ProductSection>`

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

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

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

```
562 <ProductSection>
563   <Property ovf:key="app.adminEmail" ovf:type="string" ovf:userConfigurable="true"
564     ovf:configuration="standard">
565     <Label>Admin email</Label>
566     <Description>Email address of service administrator</Description>
567   </Property>
568   <Property ovf:key="app.log" ovf:type="string" ovf:value="low"
569     ovf:userConfigurable="true">
570     <Label>Loglevel</Label>
571     <Description>Loglevel for the service</Description>
572     <Value ovf:value="none" ovf:configuration="minimal">
573   </Property>
574 </ProductSection>
```

575 In the example above, the `app.adminEmail` property is only user configurable in the standard
576 configuration, while the default value for the `app.log` property is changed from low to none in the minimal
577 configuration.

578

579 3.3.3 OperatingSystemSection element

580 The `OperatingSystemSection` element specifies the guest operating system used in a virtual system. The
581 `id` attribute unambiguously identifies the version of the operating system. The `id` attribute is required. The
582 `id` refers to an integer value from the `ValueMap` of the `CIM_OperatingSystem.OSType` property. The
583 `version` attribute contains a string mapped to the `id` in the `CIM_OperatingSystem.OSType` `Value`. Both
584 the `Info` (derived from `Section`) and `Description` elements may be externalized for localization or other
585 purposes. See section 4.2 of this document for more details on externalization. The `Version` attribute is a
586 symbolic string and cannot be internationalized.

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

```
588 <OperatingSystemSection ovf:id="76">
589   <Info>Specifies the operating system installed</Info>
590   <Description>Microsoft Windows Server 2008</Description>
591 </OperatingSystemSection>
```

592 3.3.4 EulaSection element

593 The `EulaSection` contains the human readable licensing agreement for its parent, usually a
594 `VirtualSystem` or `VirtualSystemHardware`. More than one `EulaSection` may be provided for
595 one parent. The contents of the `License` element of each `EulaSection` are displayed to the user for
596 acceptance when the OVF package is deployed. If unattended deployments are supported, provision is
597 made for implicit acceptance of the `EulaSections`.

598 Eulas may be externalized for localization or to point to an external license document. See section **Error!**
599 **Reference source not found.** of this document for more details on Internationalization.

600 This is an example `EulaSection`:

```

601 <EulaSection>
602     <Info>Licensing agreement</Info>
603     <License>
604 Lorem ipsum dolor sit amet, ligula suspendisse nulla pretium, rhoncus tempor placerat
605 fermentum, enim integer ad vestibulum volutpat. Nisl rhoncus turpis est, vel elit,
606 congue wisi enim nunc ultricies sit, magna tincidunt. Maecenas aliquam maecenas ligula
607 nostra, accumsan taciti. Sociis mauris in integer, a dolor netus non dui aliquet,
608 sagittis felis sodales, dolor sociis mauris, vel eu libero cras. Interdum at. Eget
609 habitasse elementum est, ipsum purus pede porttitor class, ut adipiscing, aliquet sed
610 auctor, imperdiet arcu per diam dapibus libero duis. Enim eros in vel, volutpat nec
611 pellentesque leo, scelerisque.
612     </License>
613 </EulaSection>

```

614 3.3.5 VirtualHardwareSection element

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

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

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

625 The following illustrates a `VirtualHardwareSection` element.

```

626 <VirtualHardwareSection>
627     <Info>Memory = 4 GB, CPU = 1 GHz, Disk = 100 GB, 1 Ethernet nic</Info>
628     <Item>
629         <rasd:AllocationUnits>Hertz*10^9</rasd:AllocationUnits>
630         <rasd:Description>Virtual CPU</rasd:Description>
631         <rasd:ElementName>1 GHz virtual CPU</rasd:ElementName>
632         <rasd:InstanceID>1</rasd:InstanceID>
633         <rasd:Reservation>1</rasd:Reservation>
634         <rasd:ResourceType>3</rasd:ResourceType>
635         <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
636         <rasd:VirtualQuantityUnit>Count</ rasd:VirtualQuantityUnit>
637     </Item>
638     <Item>
639         <rasd:AllocationUnits>byte*2^30</rasd:AllocationUnits>
640         <rasd:Description>Memory</rasd:Description>
641         <rasd:ElementName>1 GByte of memory</rasd:ElementName>
642         <rasd:InstanceID>2</rasd:InstanceID>
643         <rasd:Limit>4</rasd:Limit>
644         <rasd:Reservation>4</rasd:Reservation>
645         <rasd:ResourceType>4</rasd:ResourceType>
646     </Item>
647     <EthernetPortItem>
648         <rasd:AllocationUnits>bit / second *2^30 </rasd:AllocationUnits>

```

```

649     <epasd:Connection>VM Network</epasd:Connection>
650     <epasd:Description>Virtual NIC</epasd:Description>
651     <epasd:ElementName>Ethernet Port</epasd:ElementName>
652     <epasd:NetworkPortProfileID>1</epasd:NetworkPortProfileID>
653     <epasd:NetworkPortProfileIDType>4</epasd:NetworkPortProfileIDType>
654     <epasd:ResourceType>10</epasd:ResourceType>
655     <epasd:VirtualQuantity>1</epasd:VirtualQuantity>
656     <epasd:VirtualQuantityUnits>Count</epasd:VirtualQuantityUnits>
657 </EthernetPortItem>
658 <StorageItem>
659     <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
660     <sasd:Description>Virtual Disk</sasd:Description>
661     <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
662
663     <sasd:Reservation>100</sasd:Reservation>
664     <sasd:ResourceType>31</sasd:ResourceType>
665     <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
666     <sasd:VirtualQuantityUnit>Count</sasd:VirtualQuantityUnit>
667 </StorageItem>
668 </VirtualHardwareSection>

```

669 An example of a ResourceSubType CIM property.

```

670 <rasd:ResourceSubType>buslogic lsillogic</rasd:ResourceSubType>

```

671 The following example illustrates a VirtualHardwareSection element.

```

672 <VirtualHardwareSection>
673   <Info>...</Info>
674   <Item>
675     <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
676     <rasd:ElementName>512 MB memory size and 256 MB
677 reservation</rasd:ElementName>
678     <rasd:InstanceID>0</rasd:InstanceID>
679     <rasd:Reservation>256</rasd:Reservation>
680     <rasd:ResourceType>4</rasd:ResourceType>
681     <rasd:VirtualQuantity>512</rasd:VirtualQuantity>
682   </Item>
683   ...
684   <Item ovf:configuration="big">
685     <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
686     <rasd:ElementName>1024 MB memory size and 512 MB
687 reservation</rasd:ElementName>
688     <rasd:InstanceID>0</rasd:InstanceID>
689     <rasd:Reservation>512</rasd:Reservation>
690     <rasd:ResourceType>4</rasd:ResourceType>
691     <rasd:VirtualQuantity>1024</rasd:VirtualQuantity>
692   </Item>
693 </VirtualHardwareSection>

```

694 3.3.6 ResourceAllocationSection element

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

697 The following illustrates a ResourceAllocationSection element.

```
698 <ResourceAllocationSection>
699   <Info>Defines reservations for CPU and memory for the collection of VMs</Info>
700   <Item>
701     <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
702     <rasd:ElementName>300 MB reservation</rasd:ElementName>
703     <rasd:InstanceID>0</rasd:InstanceID>
704     <rasd:Reservation>300</rasd:Reservation>
705     <rasd:ResourceType>4</rasd:ResourceType>
706   </Item>
707   <Item ovf:configuration="..." ovf:bound="...">
708     <rasd:AllocationUnits>hertz * 10^6</rasd:AllocationUnits>
709     <rasd:ElementName>500 MHz reservation</rasd:ElementName>
710     <rasd:InstanceID>0</rasd:InstanceID>
711     <rasd:Reservation>500</rasd:Reservation>
712     <rasd:ResourceType>3</rasd:ResourceType>
713   </Item>
714   <EthernetPortItem>
715     <epasd:Address>00-16-8B-DB-00-5E</epasd:Address>
716     <epasd:Connection>VM Network</epasd:Connection>
717     <epasd:Description>Virtual NIC</epasd:Description>
718     <epasd:ElementName>Ethernet Port 1</epasd:ElementName>
719     <epasd:InstanceID>3</epasd:InstanceID>
720     <epasd:NetworkPortProfileID>1</epasd:NetworkPortProfileID>
721     <epasd:NetworkPortProfileIDType>4</epasd:NetworkPortProfileIDType>
722     <epasd:VirtualQuantityUnits>1</epasd:VirtualQuantityUnits>
723   </EthernetPortItem>
724   <StorageItem>
725     <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
726     <sasd:Description>Virtual Disk</sasd:Description>
727     <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
728     <sasd:InstanceID>4</sasd:InstanceID>
729     <sasd:Reservation>100</sasd:Reservation>
730     <sasd:ResourceType>31</sasd:ResourceType>
731     <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
732   </StorageItem>
733 </ResourceAllocationSection>
```

734 3.3.7 InstallSection element

735 The InstallSection element is optional and is used only in the VirtualSystem element. If present,
736 it is processed before the StartupSection element.

737 InstallSection elements enable the OVF author that a virtual system needs to boot before powering
738 off after installation. Typically, when the boot occurs, the guest software executes scripts or other
739 software from the OVF environment to complete the installation. The absence of an InstallSection
740 element implies a boot is not necessary to complete the installation. For example, if the virtual system has

741 no guest software or the guest software is completely installed in the system image, an
742 `InstallSection` element is not needed.

743 Several virtual systems in a virtual system collection may each have an `InstallSection` defined. In this
744 case, each virtual system is booted. The boots may be concurrent.

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

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

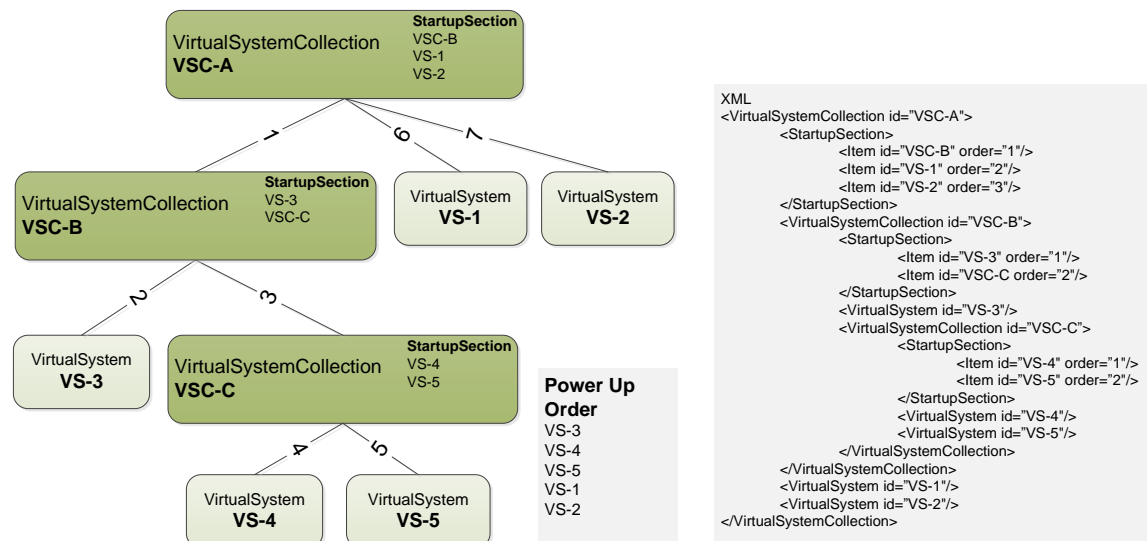
```
753 <InstallSection ovf:initialBootStopDelay="300">  
754   <Info>Specifies that the virtual machine needs to be booted after having  
755   created the guest software in order to install and/or configure the software  
756   </Info>  
757 </InstallSection>
```

758 3.3.8 StartupSection element

759 The `StartupSection` element controls powering on and off of virtual system collections and is
760 executed after `InstallSection` element(s) if any..

761 The `StartupSection` element is used in `VirtualSystemCollection` elements to specify the startup
762 and shutdown order of the virtual systems. The `StartupSection` element is a list of `Item` elements.
763 `Item` elements have attributes that control the order and timing of powering up and down. The `Item`
764 elements in a `StartupSection` element are scoped to the that element. Do not confuse these with
765 `Item` elements in a `VirtualHardwareSection` element or `ResourceAllocationSection` element.

766 The attribute in an `Item` element references the `id` attribute of an element in a `Content_Type`
767 substitution group. The `Content_Type` substitution group is a non-instantiable entity that can be
768 polymorphically replaced by either a `VirtualSystem` element or a `VirtualSystemCollection`
769 element. Thus, an `Item` element in a `StartupSection` element references either a `VirtualSystem`
770 element or a `VirtualSystemCollection` element.. A `StartupSection` element may control
771 powering up and down of both virtual systems and virtual system collections contained in the virtual
772 system collection parent. This allows for recursive start up structures. See **Figure 4**.



773

774

Figure 4 - StartupSection Traversal

775 The order of start up is determined by the value of the `order` attributes of the `Item` elements in a
 776 `StartupSection` element. The `order` attribute is a non-negative integer. If the `order` attribute of an
 777 `Item` element is zero, the virtual system or virtual system collection may be powered up at any time and
 778 the virtualization platform does not have to wait to start items with the next higher order value. Non-zero
 779 `order` attribute values are started in ascending numeric order of the order numbers starting at one. If the
 780 value of the order number is the same they may be started concurrently, but items with higher order
 781 numbers wait until lower number start.

782 Items are stopped in descending numeric order, with the exception of items with an `order` attribute value
 783 of zero, which may be stopped at any time. Items are permitted to stop in a non-descending order in an
 784 implementation specific maner unless the `shutdownorder` attribute is specified. The `shutdownorder`
 785 attribute allows the shutdown order to be specified in other than the reverse of the startup order..

786 Several optional attributes of the `Item` element support more detailed control of starting and stopping.
 787 The `startDelay` and `stopDelay` attributes specify the seconds to wait until executing the next step in
 788 the sequence. Both default to zero.

789 The `startAction` and `stopAction` attributes specify the actions to use in starting and stopping. Valid
 790 values for `startAction` are `powerOn` and `none`. The default is `powerOn`.. Valid values for the
 791 `stopAction` attribute are `powerOn`, `guestShutdown`, and `none`. The default is `powerOff`. If the
 792 `stopAction` attribute is set to `guestShutdown` the action taken is deployment platform specific.

793 The `waitingForGuest` attribute is a Boolean that allows the deployment platform to wait until the guest
 794 software reports readiness. The default value is `FALSE`. The communication mechansim is platform
 795 specific.

796 The following illustrates a `StartupSection` element.

```

797 <StartupSection>
798   <Item ovf:id="vm1" ovf:order="0" ovf:startDelay="30" ovf:stopDelay="0"
799     ovf:startAction="powerOn" ovf:waitingForGuest="true" ovf:stopAction="powerOff"/>
800   <Item ovf:id="teamA" ovf:order="0"/>
801   <Item ovf:id="vm2" ovf:order="1" ovf:startDelay="0" ovf:stopDelay="20"
802     ovf:startAction="powerOn" ovf:stopAction="guestShutdown"/>
803 </StartupSection>
  
```


804 3.3.9 EnvironmentFilesSection element

805 The `EnvironmentFilesSection` element allows the conveyance of additional environment files to the
806 guest software permitting additional customization. These files are conveyed at using the same transport
807 media as the OVF Environment file.

808 The OVF Environment file is generated by the deployment function; however these additional
809 environment files are not. The additional environment files are specified in the
810 `EnvironmentFilesSection` element with a `File` element that has an `ovf:fileRef` attribute and
811 `ovf:path` attribute for each file.

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

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

```
815 <Envelope>
816   <References>
817     ...
818   <File ovf:id="config" ovf:href="config.xml" ovf:size="4332"/>
819   <File ovf:id="resources" ovf:href="http://mywebsite/resources/resources.zip"/>
820 </References>
821   ...
822 <VirtualSystem ovf:id="...">
823   ...
824   <ovf:EnvironmentFilesSection ovf:required="false" ovf:transport="iso">
825     <Info>Config files to be included in OVF environment</Info>
826     <ovf:File ovf:fileRef="config" ovf:path="setup/cfg.xml"/>
827     <ovf:File ovf:fileRef="resources" ovf:path="setup/resources.zip"/>
828   </ovf:EnvironmentFilesSection>
829   ...
830 </VirtualSystem>
831   ...
832 </Envelope>
```

833 In the example above, the file `config.xml` in the OVF package will be copied to the OVF environment ISO
834 image and be accessible to the guest software in location `/ovffiles/setup/cfg.xml`, while the file
835 `resources.zip` will be accessible in location `/ovffiles/setup/resources.zip`.

836 3.3.10 BootDeviceSection element

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

839 a) If VM needs be setup to PXE boot from a NIC, there was no way to specify it. Similarly if VM
840 needed to be setup to boot from a secondary disk or a USB device there was no way to set that up. Thus
841 there was a need to be able to specify these alternative boot sources with their corresponding setting.

842 b) A further need was identified through implementation experience to be able to specify multiple boot
843 configurations. For instance during the “preparation” phase of the OVF, it may be necessary for a Virtual
844 System image to be patched using a fix-up disk to make it bootable.

845 The Common Information Model (CIM) defines artifacts to deal with boot order use cases prevalent in the
846 industry for BIOSes found in desktops and servers. The heart of the artifacts is `CIM_BootSourceSetting`
847 class that defines an individual boot source device like a NIC or a Disk that is the boot source. Each of
848 the devices is identified by a unique ID specified in the CIM specification.

849 A boot configuration is defined by a sequence of boot devices under an aggregation class
 850 `CIM_BootConfigSetting`. Thus a sequence of one or more `CIM_BootSourceSetting` elements is
 851 aggregated into `CIM_BootConfigSetting`.

852 The OVF envelope allows multiple such boot configurations to be aggregated into the
 853 `BootDeviceSection` element. Each such `BootDeviceSection` element can be part of a
 854 `VirtualHardwareSection` element.

855 A deployment function attempts to setup boot source sequence for a virtual system as defined in the boot
 856 configuration that it has chosen. The issue of choosing a boot configuration comes into play only when
 857 there are more than one boot configurations. The deployment function makes that choice based on the
 858 state of the deployment and the caption element in the boot configuration structure.

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

862 EXAMPLE:

```
863 <Envelope>
864 ...
865 <VirtualSystem ovf:id="...">
866 ...
867 <ovf:BootDeviceSection>
868   <Info>Boot device order specification</Info>
869   <bootc:CIM_BootConfigSetting>
870     <bootc:Caption>Pre-Install</bootc:Caption>
871     <bootc:Description>Boot Sequence for fixup of disk</bootc:Description>
872     <boots:CIM_BootSourceSetting>
873       <boots:Caption>Fix-up DVD on the network</boots:Caption>
874       <boots:InstanceID>3</boots:InstanceID>           <!-- Network device-->
875     </boots:CIM_BootSourceSetting>
876     <boots:CIM_BootSourceSetting>
877       <boots:Caption>Boot virtual disk</boots:Caption>
878       <boots:StructuredBootString>CIM:Hard-Disk</boots:StructuredBootString>
879     </boots:CIM_BootSourceSetting>
880   </bootc:CIM_BootConfigSetting>
881 </ovf:BootDeviceSection>
882 ...
883 </VirtualSystem>
884 </Envelope>
```

885 3.3.11 SharedDiskSection element

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

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

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

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

```
893 <ovf:SharedDiskSection>
894   <Info>Describes the set of virtual disks shared between VMs</Info>
895   <ovf:SharedDisk ovf:diskId="datadisk" ovf:fileRef="data">
```

```

896     ovf:capacity="8589934592" ovf:populatedSize="3549324972"
897     ovf:format="http://www.vmware.com/interfaces/specifications/vmdk.html#sparse"/>
898     <ovf:SharedDisk ovf:diskId="transientdisk" ovf:capacity="536870912"/>
899 </ovf:SharedDiskSection>

```

900 The following example illustrates the use of shared disks for Oracle Real Applications Clusters (RAC)
 901 appliance. Shared data disks are used for Oracle Clusterware and Oracle database using Oracle
 902 Automatic Storage Management (ASM). The disks for installations of Operating System (system), Oracle
 903 Clusterware (crs_home), and Oracle DB (db_home) are backed by external File references. The shared
 904 virtual disks in this example have no backing by an external File reference, the deployment engine
 905 creates the shared disk appropriately to be shared by more than one VirtualSystem.

```

906 <ovf:References>
907     <ovf:File ovf:id="system" ovf:href="system.img" ovf:compression="gzip"/>
908     <ovf:File ovf:id="crs_home" ovf:href="crs_home.img" ovf:compression="gzip"/>
909     <ovf:File ovf:id="db_home" ovf:href="db_home.img" ovf:compression="gzip"/>
910 </ovf:References>
911 <ovf:DiskSection>
912     <ovf:Info>Virtual Disks</ovf:Info>
913     <ovf:Disk ovf:diskId="system" ovf:fileRef="system" ovf:capacity="5368709120"
914     ovf:format="Raw disk image"/>
915     <ovf:Disk ovf:diskId="crs_home" ovf:fileRef="crs_home" ovf:capacity="2147483648"
916     ovf:format="Raw disk image"/>
917     <ovf:Disk ovf:diskId="db_home" ovf:fileRef="db_home" ovf:capacity="4294967296"
918     ovf:format="Raw disk image"/>
919 </ovf:DiskSection>
920 <ovf:SharedDiskSection>
921     <ovf:Info>Virtual Disks shared at runtime</ovf:Info>
922     <ovf:SharedDisk ovf:diskId="crs_asm" ovf:capacity="4294967296" ovf:format="Raw disk
923     image"/>
924     <ovf:SharedDisk ovf:diskId="db_asm" ovf:capacity="12884901888" ovf:format="Raw disk
925     image"/>
926 </ovf:SharedDiskSection>
927 .....
928 <ovf:VirtualSystemCollection ovf:id="rac_db_asm">
929     <ovf:Info>Sample Oracle RAC using ASM</ovf:Info>
930     .....
931     <ovf:ScaleOutSection ovf:id="rac_db">
932         <ovf:Info>RAC DB</ovf:Info>
933         <ovf:Description>Number of instances</ovf:Description>
934         <ovf:InstanceCount ovf:default="2" ovf:minimum="2"
935         ovf:maximum="4"></ovf:InstanceCount>
936     </ovf:ScaleOutSection>
937     .....
938     <ovf:VirtualSystem ovf:id="rac_db">
939         <ovf:Info>RAC DB Instance</ovf:Info>
940         .....
941         <ovf:VirtualHardwareSection>
942             <ovf:Info>System requirements: 8192 MB, 2 CPUs, 5 disks, 2 nics
943             </ovf:Info>
944             .....
945             <ovf:Item>
946                 <rasd:Description>Disk 1</rasd:Description>

```

```

947     <rasd:ElementName>Disk 1</rasd:ElementName>
948     <rasd:HostResource>ovf:/disk/system</rasd:HostResource>
949     <rasd:ResourceType>17</rasd:ResourceType>
950   </ovf:Item>
951   <ovf:Item>
952     <rasd:Description>Disk 2</rasd:Description>
953     <rasd:ElementName>Disk 2</rasd:ElementName>
954     <rasd:HostResource>ovf:/disk/crs_home</rasd:HostResource>
955     <rasd:ResourceType>17</rasd:ResourceType>
956   </ovf:Item>
957   <ovf:Item>
958     <rasd:Description>Disk 3</rasd:Description>
959     <rasd:ElementName>Disk 3</rasd:ElementName>
960     <rasd:HostResource>ovf:/disk/db_home</rasd:HostResource>
961     <rasd:ResourceType>17</rasd:ResourceType>
962   </ovf:Item>
963   <ovf:Item>
964     <rasd:Description>Disk 4</rasd:Description>
965     <rasd:ElementName>Disk 4</rasd:ElementName>
966     <rasd:HostResource>ovf:/disk/crs_asm</rasd:HostResource>
967     <rasd:ResourceType>17</rasd:ResourceType>
968   </ovf:Item>
969   <ovf:Item>
970     <rasd:Description>Disk 5</rasd:Description>
971     <rasd:ElementName>Disk 5</rasd:ElementName>
972     <rasd:HostResource>ovf:/disk/db_asm</rasd:HostResource>
973     <rasd:ResourceType>17</rasd:ResourceType>
974   </ovf:Item>
975   .....
976 </ovf:VirtualHardwareSection>
977 </ovf:VirtualSystem>
978 </ovf:VirtualSystemCollection>
979

```

980 3.3.12 ScaleOutSection element

981 The `ScaleOutSection` element allows dynamic configuration of the number of instantiated Virtual
982 Systems in a `Virtual System Collection` element. Without a `ScaleOut` element in the `Virtual`
983 `System Collection` element, the number of virtual systems and virtual system collections in a parent is
984 fixed. The `ScaleOutSection` element specifies a minimum and maximum number of replicas to be
985 created. At deployment time, the deployment platform chooses a value between the minimum and
986 maximum `InstanceCount`. The consumer can be queried for the value or the deployment platform can
987 make a determination based on other metadata. The `ScaleOutSection` element only appears in
988 `VirtualSystemCollection` element, although both `Virtual System` and `Virtual System Collections` may
989 be replicated.

990 The following example illustrates a `ScaleOutSection` element.

```

991 <VirtualSystemCollection ovf:id="web-tier">
992   ...
993   <ovf:ScaleOutSection ovf:id="web-server">
994     <Info>Web tier</Info>

```

```

995     <ovf:Description>Number of web server instances in web tier</ovf:Description>
996     <ovf:InstanceCount ovf:default="4" ovf:minimum="2" ovf:maximum="8"/>
997 </ovf:ScaleOutSection>
998 ...
999 <VirtualSystem ovf:id="web-server">
1000     <Info>Prototype web server</Info>
1001     ...
1002 </VirtualSystem>
1003 </VirtualSystemCollection>

```

1004 In the example above, the deployment platform creates a web tier containing between two and eight web
1005 server virtual machine instances, with a default count of four. The deployment platform makes an
1006 appropriate choice (e.g., by prompting the user). Assuming three replicas were created, the OVF
1007 environment available to the guest software in the first replica has the following content structure
1008 illustrated below:

```

1009 <Environment ... ovfenv:id="web-server-1">
1010     ...
1011     <Entity ovfenv:id="web-server-2">
1012         ...
1013     </Entity>
1014     <Entity ovfenv:id="web-server-3">
1015         ...
1016     </Entity>
1017 </Environment>

```

1018 Note that the OVF ids of the replicas are derived from the id of the prototype Virtual System by adding a
1019 sequence number. After deployment, all replica Virtual Systems will have a sequence number suffix and
1020 no Virtual System has the base id of the prototype. If there is a `StartupSection` element, then each
1021 replica has the same start up number. It is not possible to specify a start up order among replicas.

1022 EXAMPLE:

```

1023 <VirtualSystemCollection ovf:id="web-tier">
1024     ...
1025     <DeploymentOptionSection>
1026         <Info>Deployment size options</Info>
1027         <Configuration ovf:id="minimal">
1028             <Label>Minimal</Label>
1029             <Description>Minimal deployment scenario</Description>
1030         </Configuration>
1031         <Configuration ovf:id="common" ovf:default="true">
1032             <Label>Typical</Label>
1033             <Description>Common deployment scenario</Description>
1034         </Configuration>
1035         ...
1036     </DeploymentOptionSection>
1037     ...
1038     <ovf:ScaleOutSection ovf:id="web-server">
1039         <Info>Web tier</Info>
1040         <ovf:Description>Number of web server instances in web tier</ovf:Description>
1041         <ovf:InstanceCount ovf:default="4"/>
1042         <ovf:InstanceCount ovf:default="1" ovf:configuration="minimal"/>
1043     </ovf:ScaleOutSection>

```

```

1044 ...
1045 </VirtualSystemCollection>

```

In the example above, a `DeploymentOptionSection` element is used control values for `InstanceCount` in a `ScaleOutSection` element. Values in a `Scale Out Section` can also be controlled through `OVF Property` elements. Properties are prompted for one time for each replica. If the author wants a `Property` shared among replicas, the `Property` can be placed in the containing `Virtual System Collection`.

1050 3.3.13 PlacementGroupSection element

1051 The `PlacementGroupSection` defines a placement policy for a set of `VirtualSystems`. It is used
 1052 inconjunction with the `PlacementSection`.

1053 Example of `PlacementGroupSection` elements:

```

1054 <Envelope ...>
1055   ...
1056   <ovf:PlacementGroupSection ovf:id="PPid:01" ovf:policy="availability">
1057     <Info>Placement policy for group of virtual systems that need availability</Info>
1058     <ovf:Description>Placement policy for a database tier</ovf:Description>
1059   </ovf:PlacementGroupSection>
1060   ...
1061   <ovf:PlacementGroupSection ovf:id="PPid:02" ovf:policy="affinity">
1062     <Info>Placement policy for group of virtual systems that need affinity</Info>
1063     <ovf:Description>Placement policy for a web tier</ovf:Description>
1064   </ovf:PlacementGroupSection>
1065 </Envelope>

```

1066 3.3.14 PlacementSection element

1067 The `PlacementSection` specifies on a `Virtual System` or `Virtual System Collection` the placement policy as
 1068 specified in the `PlacementPolicyGroup` element to be used. This is done with an annotation of the
 1069 elements with placement policy membership id. More than one placement policy may be specified.

1070 The following illustrates a `Placement Policy` used with `Scale Out`:

```

1071 <VirtualSystemCollection ovf:id="web-tier">
1072   ...
1073   <ovf:ScaleOutSection ovf:id="web-node">
1074     <Info>Web tier</Info>
1075     ...
1076   </ovf:ScaleOutSection>
1077   ...
1078   <VirtualSystem ovf:id="web-node">
1079     <Info>Web server</Info>
1080     ...
1081     <ovf:PlacementSection ovf:group=" PPid:01">
1082       <Info>Placement policy group reference</Info>
1083     </ovf:PlacementSection>
1084     ...
1085   </VirtualSystem>
1086 </VirtualSystemCollection>

```

1087 In this example the virtual systems in the web tier should be placed on different virtualization platforms.
 1088 The placement policy applied to a `ScaleOutSection` element means that the policy is applied to each
 1089 virtual system instantiated by the deployment function.

1090 The following illustrates a Placement Policy used without a Virtual System Collection:

```

1091 <Envelope>
1092 <ovf:PlacementGroupSection ovf:id="PP20" ovf:policy="affinity ">
1093   <Info>Placement Policy for group of VMs</Info>
1094   <ovf:Description>Placement policy for VSC20</ovf:Description>
1095 </ovf:PlacementGroupSection>
1096   ...
1097   <VirtualSystem ovf:id="VSC21">
1098     <Info>Web server</Info>
1099   ...
1100     <ovf:PlacementSection ovf:group="PP20">
1101       <Info>Placement policy group reference</Info>
1102     </ovf:PlacementSection>
1103   </VirtualSystem>
1104   <VirtualSystem ovf:id="VSC22">
1105     <Info>Web server</Info>
1106   ...
1107     <ovf:PlacementSection ovf:group="PP20">
1108       <Info>Placement policy group reference</Info>
1109     </ovf:PlacementSection>
1110   </VirtualSystem>
1111   <VirtualSystem ovf:id="VSC23">
1112     <Info>Web server</Info>
1113   ...
1114     <ovf:PlacementSection ovf:group="PP20">
1115       <Info>Placement policy group reference</Info>
1116     </ovf:PlacementSection>
1117   </VirtualSystem>
1118   <VirtualSystem ovf:id="VSC24">
1119     <Info>Web server</Info>
1120   ...
1121     <ovf:PlacementSection ovf:group="PP20">
1122       <Info>Placement policy group reference</Info>
1123     </ovf:PlacementSection>
1124   </VirtualSystem>
1125 </Envelope>

```

1126 The following illustrates a Placement Policy used with nested Virtual System Collections:

```

1127 <Envelope>
1128 <ovf:PlacementGroupSection ovf:id="PP20" ovf:policy="affinity ">
1129   <Info>Placement Policy for group of VMs</Info>
1130   <ovf:Description>Placement policy for VSC20</ovf:Description>
1131 </ovf:PlacementGroupSection>
1132 <ovf:PlacementGroupSection ovf:id="PP22" ovf:policy="affinity ">
1133   <Info>Placement Policy for group of VMs</Info>
1134   <ovf:Description>Placement policy for VSC30</ovf:Description>
1135 </ovf:PlacementGroupSection>
1136 <ovf:PlacementGroupSection ovf:id="PP40" ovf:policy="availability ">
1137   <Info>Placement Policy for group of VMs</Info>
1138   <ovf:Description>Placement policy for VSC40</ovf:Description>

```

```
1139 </ovf:PlacementGroupSection>
1140     ...
1141     <VirtualSystemCollection ovf:id="VSC20">
1142 ...
1143         <ovf:PlacementSection ovf:group="PP20">
1144             <Info>Placement policy group reference</Info>
1145         </ovf:PlacementSection>
1146
1147         <VirtualSystem ovf:id=VSC21">
1148             <Info>Web server</Info>
1149 ...
1150         </VirtualSystem>
1151         <VirtualSystem ovf:id=VSC22">
1152             <Info>Web server</Info>
1153             <ovf:PlacementSection ovf:group="PP22 ">
1154                 <Info>Placement policy group reference</Info>
1155             </ovf:PlacementSection>
1156 ...
1157         </VirtualSystem>
1158         <VirtualSystem ovf:id=VSC23">
1159             <Info>Web server</Info>
1160 ...
1161         </VirtualSystem>
1162         <VirtualSystem ovf:id=VSC24">
1163             <Info>Web server</Info>
1164 ...
1165         </VirtualSystem>
1166     </VirtualSystemCollection>
1167     <VirtualSystemCollection ovf:id="VSC30">
1168 ...
1169         <ovf:PlacementSection ovf:group="PP20, PP40">
1170             <Info>Placement policy group reference</Info>
1171         </ovf:PlacementSection>
1172
1173         <VirtualSystem ovf:id=VSC31">
1174             <Info>Web server</Info>
1175 ...
1176         </VirtualSystem>
1177         <VirtualSystem ovf:id=VSC32">
1178             <Info>Web server</Info>
1179 ...
1180         </VirtualSystem>
1181         <VirtualSystem ovf:id=VSC33">
1182             <Info>Web server</Info>
1183 ...
1184         </VirtualSystem>
1185         <VirtualSystem ovf:id=VSC34">
1186             <Info>Web server</Info>
1187 ...
1188         </VirtualSystem>
```



```

1189     </VirtualSystemCollection>
1190     <VirtualSystemCollection ovf:id="VSC40">
1191     ...
1192         <ovf:PlacementSection ovf:group="PP40">
1193             <Info>Placement policy group reference</Info>
1194         </ovf:PlacementSection>
1195
1196         <VirtualSystem ovf:id=VSC41">
1197             <Info>Web server</Info>
1198             <ovf:PlacementSection ovf:group="PP22 ">
1199                 <Info>Placement policy group reference</Info>
1200             </ovf:PlacementSection>
1201         ...
1202     </VirtualSystem>
1203     <VirtualSystem ovf:id=VSC42">
1204         <Info>Web server</Info>
1205     ...
1206     </VirtualSystem>
1207     <VirtualSystem ovf:id=VSC43">
1208         <Info>Web server</Info>
1209     ...
1210     </VirtualSystem>
1211     <VirtualSystem ovf:id=VSC44">
1212         <Info>Web server</Info>
1213     ...
1214     </VirtualSystem>
1215 </VirtualSystemCollection>
1216 </Envelope>

```

1217 3.3.15 EncryptionSection element

1218 For various reasons such as licensing it is desirable to have an encryption scheme enabling free
 1219 exchange of OVF appliances while ensuring that only the intended parties can use them. The encryption
 1220 scheme proposed in this specification utilizes existing encryption standards to incorporate this
 1221 functionality in the specification,

1222 The EncryptionSection provides a single location for placing the encryption algorithm related markup and
 1223 the corresponding reference list to point to the OVF content that has been encrypted.

1224 A document would typically use a single method of encryption, with a single key. However, the
 1225 specification allows the flexibility to encrypt different portions of the OVF descriptor with different keys
 1226 derived using different methods and communicated to the end user in different ways.

1227 It is important to keep in mind that depending on which parts of the OVF descriptor have been encrypted,
 1228 an OVF descriptor may not validate against the OVF schemas until decrypted.

1229 The encryption uses XML Encryption standard 1.1 normatively to encrypt either the files in the reference
 1230 section or any parts of the XML markup of an OVF document.

1231 From an encryption standpoint, the important aspects that the standard defines are a) algorithm used for
 1232 the derivation of the key used in the encryption b) block encryption algorithm used to encrypt the content
 1233 using the key and c) method of transporting keys embedded in the OVF XML document. For each method
 1234 of encryption used within the document, all the aspects that are necessary need to be defined based on

1235 the choice of the OVF author. For instance, the author may choose to embed the key used in the
1236 document, or they may choose to communicate the key to desired end user by other means.

1237 The other aspect is a list of references to the markup sections in the OVF envelope, or the files in the
1238 reference section that are encrypted using the specific method. In order to be able to reference any
1239 section, use is made of the XML attribute named `Id` whose value of is used to point to it in the reference
1240 list.

1241 The following illustrates conceptual structure of an `Encryption` section.

```
1242 <! --- Start of encryption section ---!>
1243   <! ---- Start of Markup for encryption method 1 ----!>
1244     <! ---- Markup defining key derivation aspects per XML encryption 1.1 ----!>
1245     <! ---- Markup defining the usage of the key for encryption per XML encryption 1.1
1246     ---!>
1247     <! ---- Optionally, the markup for key transportation per XML encryption 1.1 ---!>
1248     <! ---- Start of markup for pointers to the list of XML fragments encrypted using
1249     method 1---!>
1250       <! --- Pointer 1 ---!>
1251         .
1252         .
1253       <! --- Pointer N ---!>
1254     <! ---- End of markup for pointers to the list of XML fragments encrypted using
1255     method 1 ---!>
1256     <! ---- End of Markup for method 1 of encryption ----!>
1257
1258   <! ---- Start of the markup for encryption method N ----!>
1259     <! ---- Markup defining key derivation aspects per XML encryption 1.1 ----!>
1260     <! ---- Markup defining the usage of the key for encryption per XML encryption
1261     1.1 ---!>
1262     <! ---- Optionally, the markup for key transportation per XML encryption 1.1 -
1263     --!>
1264     <! ---- Start of markup for pointers to the list of XML fragments encrypted
1265     using method 1---!>
1266       <! --- Pointer 1 ---!>
1267         .
1268         .
1269       <! --- Pointer N ---!>
1270     <! ---- End of Markup for encryption method N ----!>
1271 <! --- End of encryption section ---!>
```

1272 Below is an example of an OVF encryption section with encryption methods utilized in the OVF
1273 document, and the corresponding reference list pointing to the items that have been encrypted.

```
1274 <ovf:EncryptionSection>
1275 <!--- This section contains two different methods of encryption and the corresponding
1276 backpointers to the data that is encrypted -->
1277   <!--- Method#1: Pass phrase based Key derivation -->
1278 <!--- The following derived key block defines PBKDF2 and the corresponding back
1279 pointers to the encrypted data elements -->
1280   <!--- Use a salt value "ovfpassword" and iteration count of 4096 --->
1281   <xenc11:DerivedKey>
1282     <xenc11:KeyDerivationMethod
1283 Algorithm="http://www.rsasecurity.com/rsalabs/pkcs/schemas/pkcs-5#pbkdf2"/>
1284 <pkcs-5:PBKDF2-params>
```

```

1285         <Salt>
1286             <Specified>ovfpassword</Specified>
1287         </Salt>
1288         <IterationCount>4096</IterationCount>
1289         <KeyLength>16</KeyLength>
1290         <PRF Algorithm="http://www.w3.org/2001/04/xmlldsig-more#hmac-
1291 sha256"/>
1292     </pkcs-5:PBKDF2-params>
1293     ...
1294     <!-- The ReferenceList element below contains references to the file Ref-109.vhd via
1295     the URI syntax which is specified by XML Encryption.
1296     --->
1297     <xenc:ReferenceList>
1298         <xenc:DataReference URI="#first.vhd" />
1299     <xenc:DataReference URI=... />
1300     <xenc:DataReference URI=... />
1301     </xenc:ReferenceList>
1302     </xenc11:DerivedKey>
1303     <!-- Method#2: The following example illustrates use of a symmetric key
1304     transported using the public key within a certificate ->
1305     <xenc:EncryptedKey>
1306         <xenc:EncryptionMethod
1307             Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
1308             <ds:KeyInfo xmlns:ds='http://www.w3.org/2000/09/xmlldsig#'
1309                 <ds:X509Data>
1310                     <ds:X509Certificate> ... </ds:X509Certificate>
1311                 </ds:X509Data>
1312             </ds:KeyInfo>
1313         <xenc:CipherData>
1314             <xenc:CipherValue> ... </xenc:CipherValue>
1315         </xenc:CipherData>
1316     <!-- The ReferenceList element below contains reference #second-xml-fragment" to the
1317     XML fragment that has been encrypted using the above method --->
1318     <xenc:ReferenceList>
1319         <xenc:DataReference URI='#second-xml-fragment' />
1320         <xenc:DataReference URI='...' />
1321         <xenc:DataReference URI='...' />
1322     </xenc:ReferenceList>
1323     </xenc:EncryptedKey>
1324     </ovf:EncryptionSection>
1325     Below is an example of the encrypted file which is referenced in the EncryptionSection
1326     above using URI='Ref-109.vhd' syntax.
1327     EXAMPLE:
1328     <ovf:References>
1329     <ovf:File ovf:id="Xen:9cb10691-4012-4aeb-970c-3d47a906bfff/0b13bdba-3761-8622-22fc-
1330     2e252ed9ce14" ovf:href="Ref-109.vhd">
1331     <!-- the encrypted file referenced by the package is enclosed by an EncryptedData with
1332     a CipherReference to the actual encrypted file. The EncryptionSection in this example
1333     has a back pointer to it under the PBKDF2 algorithm via Id="first.vhd". This tells the
1334     decrypter how to decrypt the file -->
1335     <xenc:EncryptedData Id="first.vhd" Type='http://www.w3.org/2001/04/xmlenc#Element' >

```

```
1336         <xenc:EncryptionMethod
1337 Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc" />
1338         <xenc:CipherData>
1339             <xenc:CipherReference URI='Ref-109.vhd' />
1340         </xenc:CipherData>
1341 </xenc:EncryptedData>
1342 </ovf:File>
1343 </ovf:References>
```

1344 Below is an example of the encrypted OVF markup which is referenced in the
1345 EncryptionSection above using URI='#second-xml-fragment' syntax.

1346 EXAMPLE:

```
1347 <!-- the EncryptedData element below encompasses encrypted xml from the original
1348 document. It is provided with the Id "first-xml-fragment" which allows it to be
1349 referenced from the EncryptionSection. -->
1350 <xenc:EncryptedData Type=http://www.w3.org/2001/04/xmlenc#Element Id="second-xml-
1351 fragment">
1352 <!-- Each EncryptedData specifies its own encryption method. -->
1353     <xenc:EncryptionMethod Algorithm=http://www.w3.org/2001/04-xmlenc#aes128-cbc/>
1354     <xenc:CipherData>
1355         <!-- Encrypted content --->
1356         <xenc:CipherValue>DEADBEEF</xenc:CipherValue>
1357     </xenc:CipherData>
1358 </xenc:EncryptedData>
```

1359 4 Authoring an OVF package

1360 4.1 Creation

1361 The creation of an OVF package involves the i) packaging of a set of VMs onto a set of virtual disks, ii)
1362 appropriately encoding those virtual disks, iii) attaching an OVF descriptor with a specification of the
1363 virtual hardware, licensing, and other customization metadata, and iv) optionally digitally signing the
1364 package. The process of deploying an OVF package occurs when a virtualization platform consumes the
1365 OVF and creates a set of virtual machines from its contents.

1366 Creating an OVF can be made as simple as exporting an existing virtual machine from a virtualization
1367 platform into an OVF package, and adding to it the relevant metadata needed to correctly install and
1368 execute it. This transforms the virtual machine from its current runtime state on a particular hypervisor into
1369 an OVF package. During this process, the virtual machine's disks can be compressed to make it more
1370 convenient to distribute.

1371 For commercial-grade virtual appliances, a standard build environment can be used to produce an OVF
1372 package. For example, the OVF descriptor can be managed using a source control system, and the OVF
1373 package can be built using a reproducible scripting environment (such as `make` files) or, through the use
1374 of appliance building toolkits that are available from multiple vendors.

1375 When an OVF package is created, it can be accompanied with appliance-specific post-installation
1376 configuration metadata. This includes metadata for optional localization of the interface language(s) of the
1377 appliance, review/signoff and/or enforcement of the EULA, and resource configuration. It can also involve
1378 the addition of special drivers, agents and other tools to the guest to enhance (for example) I/O,
1379 timekeeping, memory management, monitoring and orderly shutdown.

1380 The process of authoring the OVF descriptor is essentially putting together the building blocks for the
1381 virtual appliance. As indicated earlier a virtual appliance is typically defined by the description of the
1382 virtual systems composing the appliance, metadata regarding the appliance and the guest OS and a set
1383 of referenced files. The OVF descriptor is the central element to aggregate and reference all required

1384 information. The major building blocks of the OVF descriptor are sections. Chapter 3 introduces the
1385 various sections which can be used to describe the virtual appliance.

1386 4.2 Internationalization

1387 The OVF specification supports localizable messages using the optional `ovf:msgid` attribute. Localized
1388 messages can be used to display the user messages in the local language during deployment.

```
1389 <Envelope ...>
1390   ...
1391   <Info ovf:msgid="info.os">Operating System</Info>
1392   ...
1393   <Strings xml:lang="da-DA">
1394     <Msg ovf:msgid="info.os">Operativsystem</Msg>
1395     ...
1396   </Strings>
1397   <Strings xml:lang="de-DE">
1398     <Msg ovf:msgid="info.os">Betriebssystem</Msg>
1399     ...
1400   </Strings>
1401 </Envelope>
```

1402 The example above defines an `Info` element within a section. The information in this section is related to
1403 the operating system of the virtual system. The attribute `ovf:msgid="info.os"` indicates that the
1404 String between start-tag and end-tag of the `Info` element can be replaced with a localized message. The
1405 localized message is referred by its message ID `info.os`. If there is a suitable localized message set in a
1406 `Strings` section, the default message “Operating System” is replaced by the localized message taken
1407 from the `Strings` section corresponding to the current region.

1408 In the example above the localized strings are stored inside the OVF descriptor. Localized strings can
1409 also be stored outside the OVF descriptor using external string bundles. For example:

```
1410 <Envelope ...>
1411   <References>
1412     ...
1413     <File ovf:id="da-DA-resources" ovf:href="danish.msg"/>
1414     <File ovf:id="de-DE-resources" ovf:href="german.msg"/>
1415     ...
1416   </References>
1417   ...
1418   <Info ovf:msgid="info.os">Operating System</Info>
1419   ...
1420   <Strings xml:lang="da-DA" ovf:fileRef="da-da-resources"/>
1421   <Strings xml:lang="de-DE" ovf:fileRef="de-de-resources"/>
1422 </Envelope>
```

1423 The localized message for “Operating System” is defined in the files `danish.msg` and `german.msg`. The
1424 format of the external message file `german.msg` is described in the example below.

```
1425 <Strings
1426   xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1"
1427   xmlns="http://schemas.dmtf.org/ovf/envelope/1"
1428   xml:lang="de-DE">
1429   ...
1430   <Msg ovf:msgid="info.os">Betriebssystem</Msg>
1431   ...
```

1432 </Strings>

1433 In the top `Strings` section the `xml:lang` attribute is used to define the locale of the particular external
 1434 message file. The external message file contains `Msg` elements for the localized messages used in the
 1435 OVF descriptor.

1436 Another method of using localized resources is to reference external files based on the current location.
 1437 This can be used to e.g. display a license text based on the location. The license text is contained in a
 1438 text file per location. The example shows how to reference an external plain text file to display a localized
 1439 license.

```

1440 <Envelope xml:lang="en-US">
1441   <References>
1442     <File ovf:id="license-en-US" ovf:href="license-en-US.txt"/>
1443     <File ovf:id="license-de-DE" ovf:href="license-de-DE.txt"/>
1444   </References>
1445   ...
1446   <VirtualSystem ovf:id="...">
1447     <EulaSection>
1448       <Info>Licensing agreement</Info>
1449       <License ovf:msgid="license">Unused</License>
1450     </EulaSection>
1451     ...
1452   </VirtualSystem>
1453   ...
1454   <Strings xml:lang="en-US">
1455     <Msg ovf:msgid="license" ovf:fileRef="license-en-US">Invalid license</Msg>
1456   </Strings>
1457   <Strings xml:lang="de-DE">
1458     <Msg ovf:msgid="license" ovf:fileRef="license-de-DE">Ihre Lizenz ist nicht
1459     gültig</Msg>
1460   </Strings>
1461 </Envelope>

```

1462 The License element contains an `ovf:msgid` attribute. In the `Strings` sections the `ovf:msgid` for the
 1463 different locations is linked to a file reference using the `ovf:fileRef` attribute. The `ovf:fileRef` attribute
 1464 has a corresponding entry in the `References` section of the OVF descriptor. The entry in the `References`
 1465 section resolves to an external text file containing the license text.

1466 4.3 Extensibility

1467 The OVF specification allows custom metadata to be added to OVF descriptors in several ways (see x.y):

- 1468 • New section elements may be defined as part of the Section substitution group, and used
 1469 wherever the OVF schemas allow sections to be present.
- 1470 • The OVF schemas use an open content model, where all existing types may be extended at the
 1471 end with additional elements. Extension points are declared in the OVF schemas with `xs:any`
 1472 declarations with `namespace="##other"`.
- 1473 • The OVF schemas allow additional attributes on existing types.

1474 A design goal of the OVF specification is to ensure backward- and forward compatibility. For forward
 1475 compatibility, this means that an OVF descriptor using features of a later specification (or custom
 1476 extensions) can be understood by an OVF consumer that is written to either i) an earlier version of the
 1477 specification, or ii) has no knowledge of the particular extensions. The OVF consumer should be able to

1478 reliably, predictably, and in a user-friendly manner, decide whether to reject or accept an OVF package
1479 that contains extensions.

1480 4.3.1 Substitution Group

1481 OVF supports an open-content model that allows additional sections to be added, as well as allowing
1482 existing sections to be extended with new content. On extensions, a Boolean `ovf:required` attribute
1483 specifies whether the information in the element is required for correct behavior or optional.

1484 Additional sections can be inserted into the OVF descriptor by defining new members of the `ovf:Section`
1485 substitution group. This means the new section extends the base schema for a `Section` element. New
1486 sections can be used to define metadata that is not related to the existing sections defined in the OVF
1487 specification. The new `Section` has an `<Info>` element that is used to display information to the
1488 consumer regarding the section in case the deployment function does not understand the section.

1489 The example shows the addition of a new `Section` `<ns:BuildInformationSection>`. The `Section` uses
1490 the namespace `ns`. The namespace is referenced in a parent element e.g. in the `<Envelope>` element:
1491 `<Envelope xmlns="http://schemas.dmtf.org/ovf/envelope/2"`
1492 `xmlns:ns="http://acme.org/ovf/extension/ns">`. As required by the `ovf:Section`
1493 substitutionGroup, the new section contains an `<Info>` element. The elements `BuildNumber`,
1494 `BuildDate`, `BuildSystem` are new elements. The elements are defined in the namespace schema
1495 referred. The `ovf:required` attribute is set to `false` to indicate that the deployment function warns but
1496 does not fail if it cannot implement the section.

1497 Example of adding new section:

```
1498 <ns:BuildInformationSection ovf:required="false">
1499   <Info>Specifies information on how a virtual machine was created</Info>
1500   <BuildNumber> ... </BuildNumber >
1501   <BuildDate> ... </BuildDate >
1502   <BuildSystem> ... </BuildSystem>
1503   ...
1504 </ns:BuildInformationSection>
```

1505 The XSD schema for the additional section in the example above looks as follows.

```
1506 <?xml version="1.0" encoding="UTF-8"?>
1507 <xs:schema xmlns:ns=http://acme.org/ovf/extension/ns
1508   xmlns:ovf=http://schemas.dmtf.org/ovf/envelope/2
1509   xmlns:xs=http://www.w3.org/2001/XMLSchema
1510   targetNamespace=http://acme.org/ovf/extension/ns
1511   elementFormDefault="qualified"
1512   attributeFormDefault="qualified">
1513 <xs:import namespace=http://schemas.dmtf.org/ovf/envelope/2
1514   schemaLocation="dsp8023.xsd"/>
1515   <xs:element name="BuildInformationSection" type="ns:BuildInformationSection_Type"
1516     substitutionGroup="ovf:Section">
1517     <xs:annotation>
1518       <xs:documentation>Element substitutable for Section since
1519       BuildInformationSection_Type is a derivation of Section_Type
1520     </xs:documentation>
1521     </xs:annotation>
1522   </xs:element>
1523   <xs:complexType name="BuildInformationSection_Type">
1524     <xs:annotation>
1525       <xs:documentation>An ACME specific section.</xs:documentation>
```

```

1526 </xs:annotation>
1527 <xs:complexContent>
1528 <xs:extension base="ovf:Section_Type">
1529 <xs:sequence>
1530 <xs:element name="BuildNumber" maxOccurs="unbounded">
1531 <xs:complexType>
1532 <xs:anyAttribute namespace="##any" processContents="lax"/>
1533 </xs:complexType>
1534 </xs:element>
1535 <xs:element name="BuildDate" maxOccurs="unbounded">
1536 <xs:complexType>
1537 <xs:anyAttribute namespace="##any" processContents="lax"/>
1538 </xs:complexType>
1539 </xs:element>
1540 <xs:element name="BuildSystem" maxOccurs="unbounded">
1541 <xs:complexType>
1542 <xs:anyAttribute namespace="##any" processContents="lax"/>
1543 </xs:complexType>
1544 </xs:element>
1545 </xs:sequence>
1546 <xs:anyAttribute namespace="##any" processContents="lax"/>
1547 </xs:extension>
1548 </xs:complexContent>
1549 </xs:complexType>
1550 </xs:schema>

```

1551 The schema defines a `BuildInformationSection` substitution group for the `ovf:Section` section. The
 1552 `BuildInformationSection` substitution group is of the `BuildInformationSection_Type` type.
 1553 `BuildInformationSection_Type` type defines `ovf:Section_Type` as a base type and extends the
 1554 `ovf:Section_Type` by the `BuildNumber`, `BuildDate` and `BuildSystem` elements.

1555 4.3.2 Elements

1556 New elements within existing sections can be added at the end of the section. The `Envelope`,
 1557 `VirtualSystem`, `VirtualSystemCollection`, `Content` and `Strings` section do not support the addition
 1558 of additional elements at the end of the section. The used namespace needs to be referenced in a parent
 1559 element and different from the OVF namespace. Additional elements can be used to extend the
 1560 information given for a particular section in the OVF descriptor.

1561 An illustration of extending an existing section is given below.

```

1562 <AnnotationSection>
1563 <Info>Specifies an annotation for this virtual machine</Info>
1564 <Annotation>This is an example of how a future element (Author) can still be
1565 parsed by older clients</Annotation>
1566 <!-- AnnotationSection extended with Author element -->
1567 <ns:Author ovf:required="false">John Smith</ns:Author>
1568 </AnnotationSection>

```

1569 The example shows an additional element in the `Annotation` section. The element extends the
 1570 `Annotation` section with information regarding the `Author` of the descriptor. The new element belongs to
 1571 the `ns` namespace.

1572 4.3.3 Attributes

1573 A third option of extending an OVF descriptor with additional information is to add custom attributes into
1574 existing elements. These attributes can be used to extend the information given by an existing element.

```
1575 <!-- Optional custom attribute example -->  
1576 <Network ovf:name="VM network" ns:desiredCapacity="1 Gbit/s">  
1577 <Description>The main network for VMs</Description>  
1578 </Network>
```

1579 The example above shows the addition of a `desiredCapacity` attribute for the `Network` element. The
1580 new attribute is defined in the `ns` namespace.

1581 See ANNEX E for more detailed examples on OVF document extensions.

1582 4.4 Conformance

1583 The OVF specification defines three conformance levels for OVF descriptors, with 1 being the highest
1584 level of conformance:

- 1585 • OVF descriptor only contains meta-data defined in the OVF specification, i.e. no custom
1586 extensions are present.
1587 Conformance Level: 1.
- 1588 • OVF descriptor contains meta-data with custom extensions, but all such extensions are
1589 optional.
1590 Conformance Level: 2.
- 1591 • OVF descriptor contains meta-data with custom extensions, and at least one such extension is
1592 required.
1593 Conformance Level: 3.

1594 The use of conformance level 3 limits portability which means that the OVF package might not be
1595 deployed on any other virtualization platform than the one supporting the custom extensions.

1596 4.5 Virtual Hardware Description

1597 The hardware description shown below is very general. In particular, it specifies that a virtual disk and a
1598 network adaptor is needed. It does not specify what the specific hardware should be. For example, a
1599 SCSI or IDE disk, or an E1000 or Vlan network card is appropriate. More specifically, it can reasonably
1600 be assumed that if the specification is generic, then the appliance will undertake discovery of the devices
1601 present, and load relevant drivers. In this case, it's assumed that the appliance creator has developed the
1602 appliance with a broad set of drivers, and has tested the appliance on relevant virtual hardware to ensure
1603 that it works.

1604 If an OVF package is deployed on a virtualization platform that does not offer the same hardware devices
1605 and/or categories of devices that are required by the guest OS that is included in the appliance, non-trivial
1606 and non-obvious installation failures can occur. The risk is not that the appliance will run incorrectly –
1607 more that it will fail to install and/or boot, and that the user will not be able to debug the problem. With this
1608 comes the risk of increased volume in customer support calls, and general customer dissatisfaction. A
1609 more constrained and detailed virtual hardware specification can reduce the chance of incorrect
1610 execution (since the specific devices required are listed) but this will limit the number of systems upon
1611 which the appliance will correctly install and/or boot.

1612 It should be borne in mind that simplicity, robustness, and predictability of installation are key reasons that
1613 ISVs are moving to the virtual appliance model, and therefore appliance developers should create
1614 appliances for which the hardware specification is more rather than less generic, unless the appliance
1615 has very specific hardware needs. At the outset, the portability of the appliance is based on the guest OS
1616 used in the virtual machines and the range of virtual hardware the guest OS supports.

1617 Ideally, the appliance vendor will create a virtual machine that has device drivers for the virtual hardware
 1618 of all of the vendor's desired target virtualization platforms. However, many virtualization platform vendors
 1619 today do not distribute drivers independently to virtual appliance vendors/creators. Instead, to further
 1620 simplify the management of the virtual hardware / appliance interface, the OVF model supports an explicit
 1621 installation mode, in which each virtual machine is booted once right after installation, to permit
 1622 localization/customization for the specific virtualization platform. This allows the virtual machine to detect
 1623 the virtualization platform and install the correct set of device drivers, including any platform specific
 1624 drivers that are made available to the guest when it first re-boots (via for example, floppy or CD drives
 1625 attached to the guest on first boot). In addition, for sysprepped Windows VMs, which need only re-
 1626 installation and customization with naming etc, the re-boot technique allows naming and tailoring of the
 1627 image to be achieved in an automated fashion.

1628 The illustration multiple virtual hardware profiles for different virtualization platforms specified in the same
 1629 descriptor.

```

1630 <VirtualHardwareSection>
1631   <Info>500Mb, 1 CPU, 1 disk, 1 nic virtual machine, Platform A</Info>
1632   <System>
1633     ...
1634   </System>
1635   <Item>
1636     ...
1637   </Item>
1638   ...
1639 </VirtualHardwareSection>
1640 <VirtualHardwareSection>
1641   <Info>500Mb, 1 CPU, 1 disk, 1 nic virtual machine, Platform B</Info>
1642   <System>
1643     ...
1644   </System>
1645   <Item>
1646     ...
1647   </Item>
1648   ...
1649 </VirtualHardwareSection>
  
```

1650 This allows the vendor to tailor the hardware description to support different virtualization platforms and
 1651 features. A specific virtualization platform may choose between any of the specific virtual hardware
 1652 sections that it can support, with the assumption that the OVF deployment function will choose the latest
 1653 or most capable feature set that is available on the local platform.

1654 The example below shows how a specific type of virtual hardware can be defined. Multiple options for the
 1655 `rasd:ResourceSubType` can be separated by a single space character. The deployment function can
 1656 then choose which virtual hardware type to instantiate.

```

1657 <Item>
1658   <rasd:ElementName>SCSI Controller 0</rasd:ElementName>
1659   <rasd:InstanceID>1000</rasd:InstanceID>
1660   <rasd:ResourceSubType>LsiLogic BusLogic</rasd:ResourceSubType>
1661   <rasd:ResourceType>6</rasd:ResourceType>
1662 </Item>
1663 <Item>
1664   <rasd:ElementName>Harddisk 1</rasd:ElementName>
1665   <rasd:HostResource>ovf:/disk/vmdisk1</rasd:HostResource>
1666   <rasd:InstanceID>22001</rasd:InstanceID>
  
```

```

1667 <rasd:Parent>1000</rasd:Parent>
1668 <rasd:ResourceType>17</rasd:ResourceType>
1669 </Item>

```

1670 4.6 Example Descriptors

1671 The following examples have been provided as complete examples of an OVF descriptor. These
 1672 examples pass XML validation.

1673 Annex A illustrates an OVF descriptor for a single virtual system.

1674 Annex B illustrates a multiple virtual system OVF descriptor.

1675 Annex C illustrates an OVF descriptor for a single virtual system with multiple applications contained in it;
 1676 i.e., a LAMP stack.

1677 Annex D illustrates an OVF descriptor for a multiple virtual system with multiple applications contained in
 1678 it, i.e., a LAMP stack with two virtual systems.

1679 5 Deploying an OVF package

1680 5.1 Deployment

1681 Deployment transforms the virtual machines in an OVF package into the runtime format understood by
 1682 the target virtualization platform, with the appropriate resource assignments and supported by the correct
 1683 virtual hardware. During deployment, the platform validates the OVF integrity, making sure that the OVF
 1684 package has not been modified in transit, and checks that it is compatible with the local virtual hardware.
 1685 It also assigns resources to, and configures the virtual machines for the particular environment on the
 1686 target virtualization platform. This includes assigning and configuring the (physical and virtual) networks
 1687 to which the virtual machines are connected; assigning storage resources for the VMs, including virtual
 1688 hard disks as well as any transient data sets, connections to clustered or networked storage and the like;
 1689 configuring CPU and memory resources, and customizing application level properties. OVF does not
 1690 support the conversion of guest software between processor architectures or hardware platforms.
 1691 Deployment instantiates one or more virtual machines with a hardware profile that is compatible with the
 1692 requirements captured in the OVF descriptor, and a set of virtual disks with the content specified in the
 1693 OVF package.

1694 The deployment experience of an OVF package depends on the virtualization platform on which it is
 1695 deployed. It could be command-line based, scripted, or a graphical deployment wizard. The typical OVF
 1696 deployment tool will show or prompt for the following information:

- 1697 • Show information about the OVF package (from the *ProductSection*), and ask the user to
 1698 accept the licensing agreement, or deal with an unattended installation.
- 1699 • Validate that the virtual hardware is compatible with the specification in the OVF.
- 1700 • Ask the user for the storage location of the virtual machines and what physical networks the
 1701 logical networks in the OVF package is connected to.
- 1702 • Ask the user to enter the specific values for the properties configured in the *ProductSection*.

1703 After this configuration, it is expected that the virtual machines can be successfully started to obtain
 1704 (using standard procedures such as DHCP) an identity that is valid on the local network. Properties are
 1705 used to prompt for specific IP network configuration and other values that are particular to the deployment
 1706 environment. Once the appliance is booted for the first time, additional configuration of software inside
 1707 the appliance can be done through a management interface provided by the appliance itself, such as a
 1708 web interface.

1709 5.2 OVF Environment Descriptor

1710 The OVF environment descriptor is an XML document that describes meta-data about the software
1711 installed on the virtual disks. The OVF specification defines the common sections used for deploying
1712 software, such as virtual hardware, disks, networks, resource requirements, and customization
1713 parameters. The descriptor is designed to be extensible so further information can be added later.

1714 A virtual appliance often needs to be customized to function properly in the particular environment where
1715 it is deployed. The OVF environment provides a standard and extensible way for the virtualization
1716 platform to communicate deployment configuration to the guest software.

1717 The OVF environment is an XML document containing deployment time customization information for the
1718 guest software. Examples of information that could be provided in the XML document include:

- 1719 • Operating system level configuration, such as host names, IP address, subnets, gateways, etc.
- 1720 • Application-level configuration such as DNS name of active directory server, databases and
1721 other external services. .

1722 The set of properties that are to be configured during deployment are specified in the OVF descriptor
1723 using the ProductSection meta-data, and is typically entered by the user using a wizard style interface
1724 during deployment.

1725 For instance, the OVF environment allows guest software to automate the network settings between
1726 multi-tiered services, and the web server may automatically configure itself with the IP address of the
1727 database server without any manual user interaction.

1728 Defining a standard OVF environment does pose some challenges, since no standard cross-vendor para-
1729 virtualized device exists for communicating between the guest software running in a virtual machine and
1730 the underlying virtualization platform. The approach taken by the OVF specification is to split the OVF
1731 environment definitions into two parts: i) A standard *protocol* that specifies what information is available
1732 and what format it is available in, and ii) a *transport*, that specifies how the information is obtained.

1733 The specification requires all implementations to support an ISO transport, which will make the OVF
1734 environment (XML document) available to the guest software on a dynamically generated ISO image.

1735 5.3 Resource Configuration Options During Deployment

1736 The OVF package has the ability to include resource configuration options for a virtual appliance. This
1737 makes it easy for the package consumer to get an initial setup without having go make individual
1738 resource decision based on the intended use. This is formatted as a human-readable list of resource
1739 configurations, for instance:

- 1740 • Software evaluation setup
- 1741 • 10-100 person workgroup setup
- 1742 • 100-1000 person workgroup setup
- 1743 • Large enterprise workgroup setup

1744 The deployment function prompts for selection of a configuration. In addition to exact values, ranges can
1745 also be specified. For example, the memory size can be specified as being 600MB, and that the
1746 recommended range is between 500MB to 1000MB. Typically, a user will not be prompted to specify a
1747 value for a range when deploying an OVF package. The list of configurations described above is
1748 expected to be used to get to a good initial resource configuration. A range specification becomes useful
1749 when the installation later needs to be changed based on different resource needs.

1750 Example list of configurations:

```
1751 <DeploymentOptionSection>
1752   <Configuration ovf:id="min">
```

```

1753     <Label>Minimal</Label>
1754     <Description>Minimal setup</Description>
1755   </Configuration>
1756   <Configuration ovf:id="normal" ovf:default="yes">
1757     <Label>Normal</Label>
1758     <Description>Standard setup</Description>
1759   </Configuration>
1760   ... more configurations ...
1761 </DeploymentOptionSection>

```

1762 **Resource requirement example:**

```

1763 <ResourceAllocationSection>
1764   <Info>Defines reservations for CPU and memory</Info>
1765   <Item>
1766     ... normal configuration ...
1767   </Item>
1768   <Item ovf:configuration="min">
1769     ... overwrites for minimal configuration ...
1770   </Item>
1771 </ResourceAllocationSection>

```

1772 **VirtualHardwareSection example:**

```

1773 <VirtualHardwareSection>
1774   <Info>...</Info>
1775   <Item>
1776     <rasd:AllocationUnits>hertz * 10^6</rasd:AllocationUnits>
1777     <rasd:ElementName>1 CPU and 500 MHz reservation</rasd:ElementName>
1778     <rasd:InstanceID>1</rasd:InstanceID>
1779     <rasd:Reservation>500</rasd:Reservation>
1780     <rasd:ResourceType>4</rasd:ResourceType>
1781     <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
1782   </Item>
1783   ...
1784   <Item ovf:configuration="big">
1785     <rasd:ElementName>1 CPU and 800 MHz reservation</rasd:ElementName>
1786     <rasd:InstanceID>0</rasd:InstanceID>
1787     <rasd:Reservation>600</rasd:Reservation>
1788     <rasd:ResourceType>3</rasd:ResourceType>
1789   </Item>
1790 </VirtualHardwareSection>

```

1791 5.4 Product Customization During Deployment Using Property Elements

1792 The OVF descriptor can contain a description of the guest software, that includes information on
 1793 customization provided through the OVF environment. This information is provided by the use of
 1794 Property elements in the ProductSection of the OVF descriptor.

1795 Each Property element has five possible attributes: `ovf:key`, `ovf:type`, `ovf:qualifiers`,
 1796 `ovf:value`, and `ovf:userConfigurable`.

1797 The `ovf:key` attribute is a unique identifier for the property element.

1798 The `ovf:type` attribute indicates the type of value the property represents.

1799 The `ovf:qualifiers` attribute specifies additional information regarding the `ovf:type` attribute so that
1800 CIM value maps can be used.

1801 The `ovf:value` attribute is used to provide a value for a `Property` element.

1802 The `ovf:userConfigurable` attribute determines if the value provided is a default value and
1803 changeable at deployment or is not changeable.

```
1804 An example of the use of Property elements follows.<ProductSection>
1805   <Info>Describes product information for the service</Info>
1806   <Product>MyService Web Portal</Product>
1807   <Vendor>Some Random Organization</Vendor>
1808   <Version>4.5</Version>
1809   <FullVersion>4.5-b4523</FullVersion>
1810   <ProductUrl>http://www.vmware.com/go/ovf</ProductUrl>
1811   <VendorUrl>http://www.vmware.com/</VendorUrl>
1812   <Property ovf:key="adminEmail" ovf:type="string" ovf:userConfigurable="true">
1813     <Description>Email address of administrator</Description>
1814   </Property>
1815   <Property ovf:key="appIp" ovf:type="string" ovf:userConfigurable="true">
1816     <Description>IP address of the application</Description>
1817   </Property>
1818   <Property ovf:key="Gateway" ovf:type="string" ovf:value="192.168.0.1"
1819   ovf:userConfigurable="false" >
1820     <Description>Gateway address to be used</Description>
1821   </Property>
1822   <Property ovf:key="ValueMap Example" ovf:type="uint8"
1823   ovf:qualifiers="uint8,uint8,string" ovf:value="1,2,three" ovf:userConfigurable="false"
1824   >
1825     <Description>Value Map Example</Description>
1826   </Property>
1827 </ProductSection>
```

1828 ANNEX D contains a detailed example of customization of a complex multi-tiered application.

1829 6 Portability

1830 OVF is an enabling technology for enhancing portability of virtual appliances and their associated virtual
1831 machines. An OVF package contains a recipe for creating virtual machines that can be interpreted
1832 concisely by a virtualization platform. The packaged meta-data enables a robust and user-friendly
1833 experience when installing a virtual appliance. In particular, the meta-data can be used by the
1834 management infrastructure to confidently decide whether a particular VM described in an OVF can be
1835 installed or whether it's rejected, and potentially to guide appropriate conversions and localizations to
1836 make it runnable in the specific execution context in which it is to be installed.

1837 There are many factors that are beyond the control of the OVF format specification and even a fully
1838 compliant implementation of it, that determine the portability of a packaged virtual machine. That is, the
1839 act of packaging a virtual machine into an OVF package does not guarantee universal portability or
1840 install-ability across all hypervisors. Below are some of the factors that could limit portability:

- 1841 • The VMs in the OVF could contain virtual disks in a format that is not understood by the
1842 hypervisor attempting the installation. While it is reasonable to expect that most hypervisors will

1843 be able to import and/or export VMs in any of the major virtual hard disk formats, newer formats
1844 may arise that are supported by the OVF and not a particular hypervisor.

1845 • The installed guest software may not support the virtual hardware presented by the hypervisor.
1846 By way of example, the Xen hypervisor does not by default offer a virtualized floppy disk device
1847 to guests. One could conceive of a guest VM that would require interaction with a floppy disk
1848 controller and which therefore would not be able to execute the VM correctly.

1849 • The installed guest software does not support the CPU architecture. For example, the guest
1850 software might execute CPU operations specific to certain processor models or require specific
1851 floating point support, or contain opcodes specific to a particular vendor's CPU.

1852 • The virtualization platform might not understand a feature requested in the OVF descriptor. For
1853 example, composed services may not be supported. Since the OVF standard will evolve
1854 independently of virtualization products, at any point an OVF might be unsupported on a
1855 virtualization platform that pre-dates that OVF specification.

1856 The portability of an OVF can be categorized into the following classes:

1857 • **Portability class 1.** Runs on multiple families of virtual hardware. For example, the appliance
1858 could be runnable on Xen, Sun, Microsoft, and VMware hypervisors. For level 3 compatibility,
1859 the guest software has been developed to support the devices of multiple hypervisors. A clean
1860 install and boot of a guest OS, during which the guest OS performs hardware device discovery
1861 and installs any specialized drivers required to interact with the virtual platform, is an example of
1862 Level 3 portability of an OVF. The "sysprep" level of portability for Microsoft Windows®
1863 operating systems is another example. Such OS instances can be re-installed, re-named and
1864 re-personalized on multiple hardware platforms, including virtual hardware.

1865 • **Portability class 2.** Runs on a specific family of virtual hardware. This would typically be due to
1866 lack of driver support by the installed guest software.

1867 • **Portability class 3.** Only runs on a particular virtualization product and/or CPU architecture
1868 and/or virtual hardware selection. This would typically be due to the OVF containing suspended
1869 virtual machines or snapshots of powered on virtual machines, including the current run-time
1870 state of the CPU and real or emulated devices. Such state ties the OVF to a very specific
1871 virtualization and hardware platform.

1872 For use within an organization, class 2 or class 3 compatibility may be good enough, since the OVF
1873 package is distributed within a controlled environment where specific purchasing decisions of hardware or
1874 virtualization platforms can ensure consistency of the underlying feature set for the OVF. A simple export
1875 of a virtual machine will typically create an OVF with class 3 or class 2 portability (tied to a specific set of
1876 virtual hardware), however it is easy to extend the metaphor to support the export of class 1 portability ,
1877 for example through the use of utilities such as "sysprep" for Windows.

1878 For commercial appliances independently created and distributed by ISVs, class 1 portability is highly
1879 desirable. Indeed, class 1 portability ensures that the appliance is readily available for the broadest
1880 possible customer base both for evaluation and production. Toolkits will generally be used to create
1881 certified "known good" class 1 packages of the appliance for broad distribution and installation on multiple
1882 virtual platforms, or class 2 portability packages if the appliance is to be consumed within the context of a
1883 narrower set of virtual hardware, such as within a particular development group in an enterprise.

1884 The OVF virtual hardware description is designed to support class 1 through class 3 portability. For class
1885 1 portability it is possible to include only very general descriptions of hardware requirements, or to specify
1886 multiple alternative virtual hardware descriptions. The appliance provider is in full control of how flexible
1887 or restrictive the virtual hardware specification is made. A narrow specification can be used to constrain
1888 an appliance to run on only known-good virtual hardware, while limiting its portability somewhat. A broad
1889 specification makes the appliance useful across as wide a set of virtual hardware as possible. This
1890 ensures that customers have the best possible user experience, which is one of the main requirements
1891 for the success of the virtual appliance concept.

ANNEX A (informative)

Single Virtual System Example

1892
1893
1894
1895

1896 Most of the descriptor is boilerplate. It starts out by describing the set of files in addition to the descriptor
1897 itself. In this case there is a single file (vmdisk1.vmdk). It then describes the set of virtual disks and the
1898 set of networks used by the appliance. Each file, disk, and network resource is given a unique identifier.
1899 These are all in separate namespaces, but the best practice is to use distinct names.

1900 The content of the example OVF is a single virtual machine. The content contains 5 sections:

1901 The following listing shows a complete OVF descriptor for a typical single virtual machine appliance:

```
1902 <?xml version="1.0" encoding="UTF-8"?>
1903 <Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1904     xmlns="http://schemas.dmtf.org/ovf/1/envelope"
1905     xmlns:ovf="http://schemas.dmtf.org/ovf/1/envelope"
1906     xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-
1907     schema/2/CIM_VirtualSystemSettingData"
1908     xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
1909     schema/2/CIM_ResourceAllocationSettingData">
1910
1911     <!-- References to all external files -->
1912     <References>
1913         <File ovf:id="file1" ovf:href="vmdisk1.vmdk" ovf:size="180114671"/>
1914     </References>
1915     <!-- Describes meta-information for all virtual disks in the package -->
1916     <DiskSection>
1917         <Info>Describes the set of virtual disks</Info>
1918         <Disk ovf:diskId="vmdisk1" ovf:fileRef="file1" ovf:capacity="4294967296"
1919         ovf:format="http://www.vmware.com/interfaces/specifications/vmdk.html#sparse"/>
1920     </DiskSection>
1921     <!-- Describes all networks used in the package -->
1922     <NetworkSection>
1923         <Info>List of logical networks used in the package</Info>
1924         <Network ovf:name="VM Network">
1925             <Description>The network that the service will be available
1926             on</Description>
1927         </Network>
1928     </NetworkSection>
1929     <VirtualSystem ovf:id="vm">
1930         <Info>Describes a virtual machine</Info>
1931         <Name>Virtual Appliance One</Name>
1932         <ProductSection>
1933             <Info>Describes product information for the appliance</Info>
1934             <Product>The Great Appliance</Product>
1935             <Vendor>Some Great Corporation</Vendor>
1936             <Version>13.00</Version>
1937             <FullVersion>13.00-b5</FullVersion>
1938         </ProductSection>

```



```

1939
1940 <ProductUrl>http://www.somegreatcorporation.com/greatappliance</ProductUrl>
1941     <VendorUrl>http://www.somegreatcorporation.com/</VendorUrl>
1942     <Property ovf:key="admin.email" ovf:type="string">
1943         <Description>Email address of administrator</Description>
1944     </Property>
1945     <Property ovf:key="app.ip" ovf:type="string"
1946     ovf:defaultValue="192.168.0.10">
1947         <Description>The IP address of this appliance</Description>
1948     </Property>
1949 </ProductSection>
1950 <AnnotationSection ovf:required="false">
1951     <Info>A random annotation on this service. It can be ignored</Info>
1952     <Annotation>Contact customer support if you have any problems</Annotation>
1953 </AnnotationSection>
1954 <EulaSection>
1955     <Info>License information for the appliance</Info>
1956     <License>Insert your favorite license here</License>
1957 </EulaSection>
1958 <VirtualHardwareSection>
1959     <Info>256MB, 1 CPU, 1 disk, 1 nic</Info>
1960     <Item>
1961         <rasd:Description>Number of virtual CPUs</rasd:Description>
1962         <rasd:ElementName>1 virtual CPU</rasd:ElementName>
1963         <rasd:InstanceID>1</rasd:InstanceID>
1964         <rasd:ResourceType>3</rasd:ResourceType>
1965         <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
1966     </Item>
1967     <Item>
1968         <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1969         <rasd:Description>Memory Size</rasd:Description>
1970         <rasd:ElementName>256 MB of memory</rasd:ElementName>
1971         <rasd:InstanceID>2</rasd:InstanceID>
1972         <rasd:ResourceType>4</rasd:ResourceType>
1973         <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
1974     </Item>
1975     <Item>
1976         <rasd:AutomaticAllocation>true</rasd:AutomaticAllocation>
1977         <rasd:Connection>VM Network</rasd:Connection>
1978         <rasd:ElementName>Ethernet adapter on "VM Network"</rasd:ElementName>
1979         <rasd:InstanceID>4000</rasd:InstanceID>
1980         <rasd:ResourceType>10</rasd:ResourceType>
1981     </Item>
1982     <Item>
1983         <rasd:ElementName>Harddisk 1</rasd:ElementName>
1984         <rasd:HostResource>ovf:/disk/vmdisk1</rasd:HostResource>
1985         <rasd:InstanceID>22001</rasd:InstanceID>
1986         <rasd:ResourceType>17</rasd:ResourceType>
1987     </Item>
1988 </VirtualHardwareSection>

```

```
1989     <OperatingSystemSection ovf:id="58" ovf:required="false">
1990         <Info>Guest Operating System</Info>
1991         <Description>Windows 2000 Advanced Server</Description>
1992     </OperatingSystemSection>
1993 </VirtualSystem>
1994 </Envelope>
```

1995

**ANNEX B
(informative)**

Multi-tiered Pet Store Example

1996
1997
1998
1999

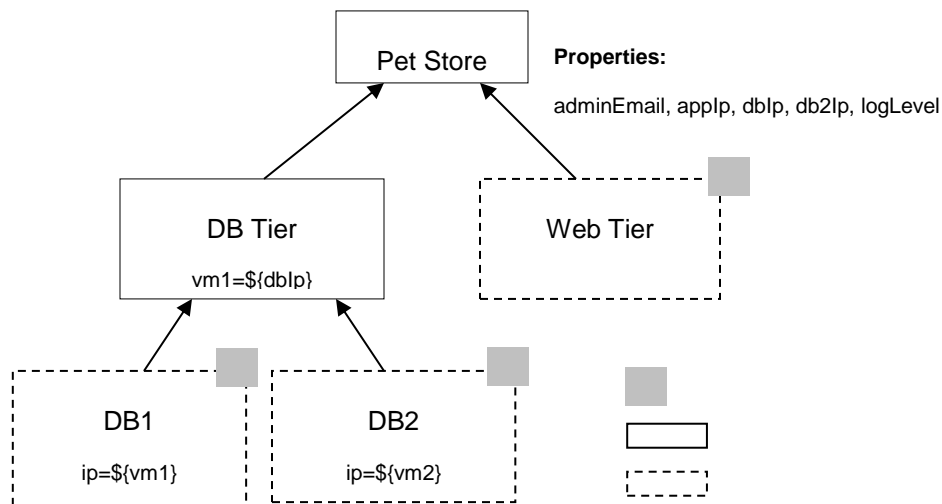
2000 This example will demonstrate several advanced OVF concepts:

- 2001 • Multi-VM packages - use of the VirtualMachineCollection entity subtype
- 2002 • Composite service organization - use of nested VirtualMachineCollection entity subtype
- 2003 • Propagation of user defined deployment configuration.
- 2004 • Deployment time customization of the service using the OVF Environment.
- 2005 • The use of virtual disk chains to minimize downloads.
- 2006 • Nesting of ProductSections for providing information about the installed software in an individual
- 2007 virtual machine

2008 The example service is called Pet Store and consists of a front-end web-server and a database. The
2009 database server is itself a complex multi-tiered server consisting of two VMs for fault-tolerance.

2010 B.1 Architecture and Packaging

2011 The Pet Store OVF package consists of 3 virtual systems (WebTier, DB1, and DB2) and 2 virtual system
2012 collections (Pet Store and DBTier). The diagram below shows the structure of the OVF package as well
2013 as the properties and startup order of the virtual machines:



2014
2015

Figure 5 – Pet Store OVF Package

2016 The complete OVF descriptor is listed at the end of this document. The use of properties and disk layout
2017 of the OVF is discussed in more details in the following.

2018 **B.2 Properties**

2019 The Pet Store service has 5 user-configurable properties. These are the key control parameters for the
 2020 service that needs to be configured in order for it to start up correctly in the deployed environment. The
 2021 properties are passed up to the guest software in the form of an OVF environment document. The guest
 2022 software is written to read the OVF environment on startup, extract the values of the properties, and apply
 2023 them to the software configuration. Thus, the OVF descriptor reflects the properties that are handled by
 2024 the guest software.

2025 For this particular service, there are two different software configurations, one for the Web tier and one for
 2026 the Database tier. The properties supported in each software configuration are:

2027 **Table 1** illustrates the properties for the Web Guest Software:

2028 **Table 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 connect to.
<i>adminEmail</i>	Email address for support
<i>logLevel</i>	Logging level

2029

2030 All properties defined on the immediate parent VirtualSystemCollection container is available to a child
 2031 VirtualSystem or VirtualSystemCollection. Thus, the OVF descriptor does not need to contain an explicit
 2032 ProductSection for each VM, as demonstrated for WebVM.

2033 **Table 2** illustrates the properties for the Database Guest Software:

2034 **Table 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>	Here the logging level is called log

2035 The clustered database is organized as a virtual system collection itself with a specific set of properties
 2036 for configuration: vm1, vm2, and log. This organization separates the database implementation from the
 2037 rest of the software in the OVF package and allows virtual appliances (guest software + virtual machine
 2038 configurations) to be easily composed and thereby promotes reuse.

2039 The database software is an off-the-shelf software package and the vendor has chosen the
 2040 "com.mydb.db" as the unique name for all the properties. This can be seen in the OVF descriptor with the
 2041 inclusion of the ovf:class attribute on the ProductSection.

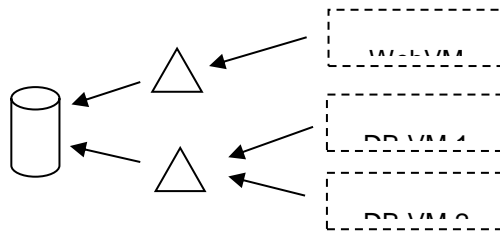
2042 The \${<name>} property syntax is used to propagate values from the outer level into the inner nodes in
 2043 the OVF Descriptor's entity hierarchy. This mechanism allows linking up different components without
 2044 having to pre-negotiate naming conventions or changing guest software. Only properties defined on the
 2045 immediate parent VirtualSystemCollection container are available to a child entity. Thus, properties

2046 defined on Petstore will not be available to a DB1. This ensures that the interface for a
 2047 VirtualSystemCollection is encapsulated and well described in its parent VirtualSystemCollection, which
 2048 makes the software composable and easy to reuse.

2049 The OVF descriptor uses fixed non-user assignable properties to ensure that the two database virtual
 2050 machines boots up into different roles even though they are, initially, booting of the exact same software
 2051 image. The property named *com.mydb.db.primaryAtBoot* is specified with a fixed, non-user configurable
 2052 value but is different value for the two images. The software inspects this at boot time and customizes its
 2053 operation accordingly.

2054 B.3 Disk Layout

2055 The Petstore OVF package uses the ability to share disks and encode a delta disk hierarchy to minimize
 2056 the size and thereby the download time for the package. In this particular case, we only have two different
 2057 images (Database and Web), and if we further assume they are build on top of the same base OS
 2058 distribution, we can encode this in the OVF descriptor as.



2059

2060

Figure 6 – Pet Store Virtual Disk Layout

2061 Thus, while the package contains 3 distinct virtual machines, the total download size will be significantly
 2062 smaller. In fact, only one full VM and then two relative small deltas need to be downloaded.

2063 The physical layout of the virtual disks on the deployment system is independent of the disk structure in
 2064 the OVF package. The OVF package describes the size of the virtual disk and the content (i.e., bits that
 2065 needs to be on the disk). It also specifies that each virtual machine gets independent disks. Thus, a
 2066 virtualization platform could install the above package as a 3 VMs with 3 independent flat disks, or it could
 2067 chose to replicate the above organization, or something third, as long as each virtual machine sees a disk
 2068 with the content described on initial boot and that changes written by one virtual machine does not affect
 2069 the others.

2070 B.4 Pet Store OVF Descriptor

```

2071 <?xml version="1.0" encoding="UTF-8"?>
2072 <Envelope
2073   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2074   xmlns="http://schemas.dmtf.org/ovf/envelope/1"
2075   xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1"
2076   xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2077 schema/2/CIM_VirtualSystemSettingData"
2078   xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2079 schema/2/CIM_ResourceAllocationSettingData">
2080   <!-- References to all external files -->
2081   <References>
2082     <File ovf:id="base" ovf:href="base.vmdk" ovf:size="180114671"/>
2083     <File ovf:id="webdelta" ovf:href="webapp-delta.vmdk" ovf:size="123413"/>
2084     <File ovf:id="dbdelta" ovf:href="dbapp-delta.vmdk" ovf:size="343243"/>
2085   </References>
2086   <!-- Describes meta-information about all virtual disks in the package.
  
```

```

2087     This example is encoded as a delta-disk hierarchy.
2088     -->
2089     <DiskSection>
2090         <Info>Describes the set of virtual disks</Info>
2091         <Disk ovf:diskId="base" ovf:fileRef="base" ovf:capacity="4294967296"
2092             ovf:populatedSize="1924967692"
2093
2094         ovf:format="http://www.vmware.com/specifications/vmdk.html#streamOptimized"/>
2095         <Disk ovf:diskId="web" ovf:fileRef="webappdelta" ovf:parentRef="base"
2096             ovf:capacity="4294967296"
2097
2098         ovf:format="http://www.vmware.com/specifications/vmdk.html#streamOptimized"/>
2099         <Disk ovf:diskId="db" ovf:fileRef="dbdelta" ovf:parentRef="base"
2100             ovf:capacity="4294967296"
2101
2102         ovf:format="http://www.vmware.com/specifications/vmdk.html#streamOptimized"/>
2103     </DiskSection>
2104     <!-- Describes all networks used in the package -->
2105     <NetworkSection>
2106         <Info>List of logical networks used in the package</Info>
2107         <Network ovf:name="VM Network">
2108             <Description ovf:msgid="network.description">The network that the service
2109                 will be available on</Description>
2110         </Network>
2111     </NetworkSection>
2112     <!-- Deployment options for the packages -->
2113     <DeploymentOptionSection>
2114         <Info>List of deployment options available in the package</Info>
2115         <Configuration ovf:id="minimal">
2116             <Label ovf:msgid="minimal.label">Minimal</Label>
2117             <Description ovf:msgid="minimal.description">Deploy service with minimal
2118                 resource use</Description>
2119         </Configuration>
2120         <Configuration ovf:id="standard" ovf:default="true">
2121             <Label ovf:msgid="standard.label">Standard</Label>
2122             <Description ovf:msgid="standard.description">Deploy service with standard
2123                 resource use</Description>
2124         </Configuration>
2125     </DeploymentOptionSection>
2126     <!-- PetStore Virtual System Collection -->
2127     <VirtualSystemCollection ovf:id="PetStore">
2128         <Info>The packaging of the PetStoreService multi-tier application</Info>
2129         <Name>PetStore Service</Name>
2130         <!-- Overall information about the product -->
2131         <ProductSection>
2132             <Info>Describes product information for the service</Info>
2133             <Product>PetStore Web Portal</Product>
2134             <Vendor>Some Random Organization</Vendor>
2135             <Version>4.5</Version>
2136             <FullVersion>4.5-b4523</FullVersion>
2137             <ProductUrl>http://www.vmware.com/go/ovf</ProductUrl>
2138             <VendorUrl>http://www.vmware.com/</VendorUrl>
2139             <Category ovf:msgid="category.email">Email properties</Category>
2140             <Property ovf:key="adminEmail" ovf:type="string"
2141                 ovf:userConfigurable="true">
2142                 <Label ovf:msgid="property.email.label">Admin email</Label>
2143                 <Description ovf:msgid="property.email.description">Email address of
2144                     service administrator</Description>
2145             </Property>
2146             <Category ovf:msgid="category.network">Network properties</Category>
2147             <Property ovf:key="appIp" ovf:type="string"
2148                 ovf:userConfigurable="true">
2149                 <Label ovf:msgid="property.appip.label">IP</Label>
2150                 <Description ovf:msgid="property.appip.description">IP address of the

```

```

2151         service</Description>
2152     </Property>
2153     <Property ovf:key="dbIp" ovf:type="string" ovf:userConfigurable="true">
2154         <Label ovf:msgid="property.dpip.label">IP for DB</Label>
2155         <Description ovf:msgid="property.dpip.description">Primary IP address
of
2156
2157         the database</Description>
2158     </Property>
2159     <Property ovf:key="db2Ip" ovf:type="string"
2160         ovf:userConfigurable="true">
2161         <Label ovf:msgid="property.dpip2.label">IP for DB2</Label>
2162         <Description ovf:msgid="property.dpip2.description">A secondary IP
2163         address for the database</Description>
2164     </Property>
2165     <Category ovf:msgid="category.logging">Logging properties</Category>
2166     <Property ovf:key="logLevel" ovf:type="string" ovf:value="normal"
2167         ovf:userConfigurable="true">
2168         <Label ovf:msgid="property.loglevel.label">Loglevel</Label>
2169         <Description ovf:msgid="property.loglevel.description">Logging level
for
2170
2171         the service</Description>
2172         <Value ovf:value="low" ovf:configuration="minimal"/>
2173     </Property>
2174 </ProductSection>
2175 <AnnotationSection ovf:required="false">
2176     <Info>A annotation on this service</Info>
2177     <Annotation ovf:msgid="annotation.annotation">Contact customer support for
2178     any urgent issues</Annotation>
2179 </AnnotationSection>
2180 <ResourceAllocationSection ovf:required="false">
2181     <Info>Defines minimum reservations for CPU and memory</Info>
2182     <Item>
2183         <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
2184         <rasd:ElementName>512 MB reservation</rasd:ElementName>
2185         <rasd:InstanceID>0</rasd:InstanceID>
2186         <rasd:Reservation>512</rasd:Reservation>
2187         <rasd:ResourceType>4</rasd:ResourceType>
2188     </Item>
2189     <Item ovf:configuration="minimal">
2190         <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
2191         <rasd:ElementName>384 MB reservation</rasd:ElementName>
2192         <rasd:InstanceID>0</rasd:InstanceID>
2193         <rasd:Reservation>384</rasd:Reservation>
2194         <rasd:ResourceType>4</rasd:ResourceType>
2195     </Item>
2196     <Item>
2197         <rasd:AllocationUnits>MHz</rasd:AllocationUnits>
2198         <rasd:ElementName>1000 MHz reservation</rasd:ElementName>
2199         <rasd:InstanceID>1</rasd:InstanceID>
2200         <rasd:Reservation>500</rasd:Reservation>
2201         <rasd:ResourceType>3</rasd:ResourceType>
2202     </Item>
2203     <Item ovf:bound="min">
2204         <rasd:AllocationUnits>MHz</rasd:AllocationUnits>
2205         <rasd:ElementName>500 MHz reservation</rasd:ElementName>
2206         <rasd:InstanceID>1</rasd:InstanceID>
2207         <rasd:Reservation>500</rasd:Reservation>
2208         <rasd:ResourceType>3</rasd:ResourceType>
2209     </Item>
2210     <Item ovf:bound="max">
2211         <rasd:AllocationUnits>MHz</rasd:AllocationUnits>
2212         <rasd:ElementName>1500 MHz reservation</rasd:ElementName>
2213         <rasd:InstanceID>1</rasd:InstanceID>
2214         <rasd:Reservation>1500</rasd:Reservation>

```

```

2215         <rasd:ResourceType>3</rasd:ResourceType>
2216     </Item>
2217 </ResourceAllocationSection>
2218 <StartupSection>
2219     <Info>Specifies how the composite service is powered-on and off</Info>
2220     <Item ovf:id="DBTier" ovf:order="1" ovf:startDelay="120"
2221         ovf:startAction="powerOn" ovf:waitingForGuest="true"
2222 ovf:stopDelay="120"
2223         ovf:stopAction="guestShutdown"/>
2224     <Item ovf:id="WebTier" ovf:order="2" ovf:startDelay="120"
2225         ovf:startAction="powerOn" ovf:waitingForGuest="true"
2226 ovf:stopDelay="120"
2227         ovf:stopAction="guestShutdown"/>
2228 </StartupSection>
2229 <VirtualSystem ovf:id="WebTier">
2230     <Info>The virtual machine containing the WebServer application</Info>
2231     <ProductSection>
2232         <Info>Describes the product information</Info>
2233         <Product>Apache Webserver</Product>
2234         <Vendor>Apache Software Foundation</Vendor>
2235         <Version>6.5</Version>
2236         <FullVersion>6.5-b2432</FullVersion>
2237     </ProductSection>
2238     <OperatingSystemSection ovf:id="97">
2239         <Info>Guest Operating System</Info>
2240         <Description>Linux 2.4.x</Description>
2241     </OperatingSystemSection>
2242     <VirtualHardwareSection>
2243         <Info>256 MB, 1 CPU, 1 disk, 1 nic virtual machine</Info>
2244         <System>
2245             <vssd:ElementName>Virtual Hardware Family</vssd:ElementName>
2246             <vssd:InstanceID>0</vssd:InstanceID>
2247             <vssd:VirtualSystemType>vmx-04</vssd:VirtualSystemType>
2248         </System>
2249         <Item>
2250             <rasd:Description>Number of virtual CPUs</rasd:Description>
2251             <rasd:ElementName>1 virtual CPU</rasd:ElementName>
2252             <rasd:InstanceID>1</rasd:InstanceID>
2253             <rasd:ResourceType>3</rasd:ResourceType>
2254             <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2255         </Item>
2256         <Item>
2257             <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
2258             <rasd:Description>Memory Size</rasd:Description>
2259             <rasd:ElementName>256 MB of memory</rasd:ElementName>
2260             <rasd:InstanceID>2</rasd:InstanceID>
2261             <rasd:ResourceType>4</rasd:ResourceType>
2262             <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
2263         </Item>
2264         <Item>
2265             <rasd:AutomaticAllocation>true</rasd:AutomaticAllocation>
2266             <rasd:Connection>VM Network</rasd:Connection>
2267             <rasd:ElementName>Ethernet adapter on "VM
2268 Network"</rasd:ElementName>
2269             <rasd:InstanceID>3</rasd:InstanceID>
2270             <rasd:ResourceSubType>PCNet32</rasd:ResourceSubType>
2271             <rasd:ResourceType>10</rasd:ResourceType>
2272         </Item>
2273         <Item>
2274             <rasd:AddressOnParent>1</rasd:AddressOnParent>
2275             <rasd:ElementName>SCSI Controller 0 - LSI Logic</rasd:ElementName>
2276             <rasd:InstanceID>1000</rasd:InstanceID>
2277             <rasd:ResourceSubType>LsiLogic</rasd:ResourceSubType>
2278             <rasd:ResourceType>6</rasd:ResourceType>

```



```

2279         </Item>
2280     <Item>
2281         <rasd:AddressOnParent>0</rasd:AddressOnParent>
2282         <rasd:ElementName>Harddisk 1</rasd:ElementName>
2283         <rasd:HostResource>ovf:/disk/web</rasd:HostResource>
2284         <rasd:InstanceID>22001</rasd:InstanceID>
2285         <rasd:Parent>1000</rasd:Parent>
2286         <rasd:ResourceType>17</rasd:ResourceType>
2287     </Item>
2288 </VirtualHardwareSection>
2289 </VirtualSystem>
2290 <!-- Database Tier -->
2291 <VirtualSystemCollection ovf:id="DBTier">
2292     <Info>Describes a clustered database instance</Info>
2293     <ProductSection ovf:class="com.mydb.db">
2294         <Info>Product Information</Info>
2295         <Product>Somebody Clustered SQL Server</Product>
2296         <Vendor>TBD</Vendor>
2297         <Version>2.5</Version>
2298         <FullVersion>2.5-bl234</FullVersion>
2299         <Property ovf:key="vm1" ovf:value="{dbIp}" ovf:type="string"/>
2300         <Property ovf:key="vm2" ovf:value="{db2Ip}" ovf:type="string"/>
2301         <Property ovf:key="log" ovf:value="{logLevel}" ovf:type="string"/>
2302     </ProductSection>
2303     <StartupSection>
2304         <Info>Specifies how the composite service is powered-on and off</Info>
2305         <Item ovf:id="DB1" ovf:order="1" ovf:startDelay="120"
2306             ovf:startAction="powerOn" ovf:waitingForGuest="true"
2307             ovf:stopDelay="120" ovf:stopAction="guestShutdown"/>
2308         <Item ovf:id="DB2" ovf:order="2" ovf:startDelay="120"
2309             ovf:startAction="powerOn" ovf:waitingForGuest="true"
2310             ovf:stopDelay="120" ovf:stopAction="guestShutdown"/>
2311     </StartupSection>
2312     <!-- DB VM 1 -->
2313     <VirtualSystem ovf:id="DB1">
2314         <Info>Describes a virtual machine with the database image
2315 installed</Info>
2316         <Name>Database Instance I</Name>
2317         <ProductSection ovf:class="com.mydb.db">
2318             <Info>Specifies the OVF properties available in the OVF
2319 environment</Info>
2320             <Property ovf:key="ip" ovf:value="{vm1}" ovf:type="string"/>
2321             <Property ovf:key="ip2" ovf:value="{vm2}" ovf:type="string"/>
2322             <Property ovf:key="primaryAtBoot" ovf:value="yes"
2323 ovf:type="string"/>
2324         </ProductSection>
2325         <VirtualHardwareSection>
2326             <Info>256 MB, 1 CPU, 1 disk, 1 nic virtual machine</Info>
2327             <System>
2328                 <vssd:ElementName>Virtual Hardware Family</vssd:ElementName>
2329                 <vssd:InstanceID>0</vssd:InstanceID>
2330                 <vssd:VirtualSystemType>vmx-04</vssd:VirtualSystemType>
2331             </System>
2332             <Item>
2333                 <rasd:Description>Number of virtual CPUs</rasd:Description>
2334                 <rasd:ElementName>1 virtual CPU</rasd:ElementName>
2335                 <rasd:InstanceID>1</rasd:InstanceID>
2336                 <rasd:ResourceType>3</rasd:ResourceType>
2337                 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2338             </Item>
2339             <Item>
2340                 <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
2341                 <rasd:Description>Memory Size</rasd:Description>
2342                 <rasd:ElementName>256 MB of memory</rasd:ElementName>

```

```

2343         <rasd:InstanceID>2</rasd:InstanceID>
2344         <rasd:ResourceType>4</rasd:ResourceType>
2345         <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
2346     </Item>
2347     <Item>
2348         <rasd:AutomaticAllocation>true</rasd:AutomaticAllocation>
2349         <rasd:Connection>VM Network</rasd:Connection>
2350         <rasd:ElementName>Ethernet adapter on "VM
2351 Network"</rasd:ElementName>
2352         <rasd:InstanceID>3</rasd:InstanceID>
2353         <rasd:ResourceSubType>PCNet32</rasd:ResourceSubType>
2354         <rasd:ResourceType>10</rasd:ResourceType>
2355     </Item>
2356     <Item>
2357         <rasd:AddressOnParent>1</rasd:AddressOnParent>
2358         <rasd:ElementName>SCSI Controller 0 - LSI
2359 Logic</rasd:ElementName>
2360         <rasd:InstanceID>1000</rasd:InstanceID>
2361         <rasd:ResourceSubType>LsiLogic</rasd:ResourceSubType>
2362         <rasd:ResourceType>6</rasd:ResourceType>
2363     </Item>
2364     <Item>
2365         <rasd:AddressOnParent>0</rasd:AddressOnParent>
2366         <rasd:ElementName>Harddisk 1</rasd:ElementName>
2367         <rasd:HostResource>ovf:/disk/db</rasd:HostResource>
2368         <rasd:InstanceID>22001</rasd:InstanceID>
2369         <rasd:Parent>1000</rasd:Parent>
2370         <rasd:ResourceType>17</rasd:ResourceType>
2371     </Item>
2372 </VirtualHardwareSection>
2373 <OperatingSystemSection ovf:id="97">
2374     <Info>Guest Operating System</Info>
2375     <Description>Linux 2.4.x</Description>
2376 </OperatingSystemSection>
2377 </VirtualSystem>
2378 <!-- DB VM 2 -->
2379 <VirtualSystem ovf:id="DB2">
2380     <Info>Describes a virtual machine with the database image
2381 installed</Info>
2382     <Name>Database Instance II</Name>
2383     <ProductSection ovf:class="com.mydb.db">
2384         <Info>Specifies the OVF properties available in the OVF
2385 environment</Info>
2386         <Property ovf:key="ip" ovf:value="{vm2}" ovf:type="string"/>
2387         <Property ovf:key="ip2" ovf:value="{vm1}" ovf:type="string"/>
2388         <Property ovf:key="primaryAtBoot" ovf:value="no"
2389 ovf:type="string"/>
2390     </ProductSection>
2391 </VirtualHardwareSection>
2392     <Info>256 MB, 1 CPU, 1 disk, 1 nic virtual machine</Info>
2393     <System>
2394         <vssd:ElementName>Virtual Hardware Family</vssd:ElementName>
2395         <vssd:InstanceID>0</vssd:InstanceID>
2396         <vssd:VirtualSystemType>vmx-04</vssd:VirtualSystemType>
2397     </System>
2398     <Item>
2399         <rasd:Description>Number of virtual CPUs</rasd:Description>
2400         <rasd:ElementName>1 virtual CPU</rasd:ElementName>
2401         <rasd:InstanceID>1</rasd:InstanceID>
2402         <rasd:ResourceType>3</rasd:ResourceType>
2403         <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2404     </Item>
2405     <Item>
2406         <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>

```

```

2407         <rasd:Description>Memory Size</rasd:Description>
2408         <rasd:ElementName>256 MB of memory</rasd:ElementName>
2409         <rasd:InstanceID>2</rasd:InstanceID>
2410         <rasd:ResourceType>4</rasd:ResourceType>
2411         <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
2412     </Item>
2413     <Item>
2414         <rasd:AutomaticAllocation>true</rasd:AutomaticAllocation>
2415         <rasd:Connection>VM Network</rasd:Connection>
2416         <rasd:ElementName>Ethernet adapter on "VM
2417 Network"</rasd:ElementName>
2418         <rasd:InstanceID>3</rasd:InstanceID>
2419         <rasd:ResourceSubType>PCNet32</rasd:ResourceSubType>
2420         <rasd:ResourceType>10</rasd:ResourceType>
2421     </Item>
2422     <Item>
2423         <rasd:AddressOnParent>1</rasd:AddressOnParent>
2424         <rasd:ElementName>SCSI Controller 0 - LSI
2425 Logic</rasd:ElementName>
2426         <rasd:InstanceID>1000</rasd:InstanceID>
2427         <rasd:ResourceSubType>LsiLogic</rasd:ResourceSubType>
2428         <rasd:ResourceType>6</rasd:ResourceType>
2429     </Item>
2430     <Item>
2431         <rasd:AddressOnParent>0</rasd:AddressOnParent>
2432         <rasd:ElementName>Harddisk 1</rasd:ElementName>
2433         <rasd:HostResource>ovf:/disk/db</rasd:HostResource>
2434         <rasd:InstanceID>22001</rasd:InstanceID>
2435         <rasd:Parent>1000</rasd:Parent>
2436         <rasd:ResourceType>17</rasd:ResourceType>
2437     </Item>
2438 </VirtualHardwareSection>
2439 <OperatingSystemSection ovf:id="97">
2440     <Info>Guest Operating System</Info>
2441     <Description>Linux 2.4.x</Description>
2442 </OperatingSystemSection>
2443 </VirtualSystem>
2444 </VirtualSystemCollection>
2445 </VirtualSystemCollection>
2446 <!-- External I18N bundles -->
2447 <Strings xml:lang="de-DE" ovf:fileRef="de-DE-bundle.xml"/>
2448 <!-- Embedded I18N bundles -->
2449 <Strings xml:lang="da-DA">
2450     <Msg ovf:msgid="network.description">Netværket servicen skal være tilgængelig
2451 på</Msg>
2452     <Msg ovf:msgid="annotation.annotation">Kontakt kundeservice i tilfælde af
2453     kritiske problemer</Msg>
2454     <Msg ovf:msgid="property.email.description">Email adresse for
2455 administrator</Msg>
2456     <Msg ovf:msgid="property.appip.description">IP adresse for service</Msg>
2457     <Msg ovf:msgid="property.dpip">Primær IP adresse for database</Msg>
2458     <Msg ovf:msgid="property.dpip2.description">Sekundær IP adresse for
2459 database</Msg>
2460     <Msg ovf:msgid="property.loglevel.description">Logningsniveau for
2461 service</Msg>
2462     <Msg ovf:msgid="minimal.label">Minimal</Msg>
2463     <Msg ovf:msgid="minimal.description">Installer service med minimal brug af
2464     resourcer</Msg>
2465     <Msg ovf:msgid="standard.label">Normal</Msg>
2466     <Msg ovf:msgid="standard.description">Installer service med normal brug af
2467     resourcer</Msg>
2468 </Strings>
2469 </Envelope>

```

2470 **B.5 Complete OVF Environment**

2471 The following lists the OVF environments seen by the WebTier and DB1 virtual machines (DB2 is is
2472 virtually identical to the one for DB1 and is omitted).

2473 OVF environment for the WebTier virtual machine:

```
2474 <?xml version="1.0" encoding="UTF-8"?>
2475 <Environment
2476     xmlns="http://schemas.dmtf.org/ovf/environment/1"
2477     xmlns:ovfenv="http://schemas.dmtf.org/ovf/environment/1"
2478     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2479     ovfenv:id="WebTier">
2480
2481     <!-- Information about hypervisor platform -->
2482     <PlatformSection>
2483         <Kind>ESX Server</Kind>
2484         <Version>3.0.1</Version>
2485         <Vendor>VMware, Inc.</Vendor>
2486         <Locale>en_US</Locale>
2487     </PlatformSection>
2488
2489     <!-- Properties defined for this virtual machine -->
2490     <PropertySection>
2491         <Property ovfenv:key="adminEmail" ovfenv:value="ovf-admin@vmware.com"/>
2492         <Property ovfenv:key="appIp" ovfenv:value="10.20.132.101"/>
2493         <Property ovfenv:key="dbIp" ovfenv:value="10.20.132.102"/>
2494         <Property ovfenv:key="db2Ip" ovfenv:value="10.20.132.103"/>
2495         <Property ovfenv:key="logLevel" ovfenv:value="warning"/>
2496     </PropertySection>
2497
2498     <Entity ovfenv:id="DBTier">
2499         <PropertySection>
2500             <Property ovfenv:key="adminEmail" ovfenv:value="ovf-admin@vmware.com"/>
2501             <Property ovfenv:key="appIp" ovfenv:value="10.20.132.101"/>
2502             <Property ovfenv:key="dbIp" ovfenv:value="10.20.132.102"/>
2503             <Property ovfenv:key="db2Ip" ovfenv:value="10.20.132.103"/>
2504             <Property ovfenv:key="logLevel" ovfenv:value="warning"/>
2505             <Property ovfenv:key="com.mydb.db.vm1" ovfenv:value="10.20.132.102"/>
2506             <Property ovfenv:key="com.mydb.db.vm2" ovfenv:value="10.20.132.103"/>
2507             <Property ovfenv:key="com.mydb.db.log" ovfenv:value="warning"/>
2508         </PropertySection>
2509     </Entity>
2510 </Environment>
```

2511 OVF environment for the DB1 virtual machine:

```
2512 <?xml version="1.0" encoding="UTF-8"?>
2513 <Environment
2514     xmlns="http://schemas.dmtf.org/ovf/environment/1"
2515     xmlns:ovfenv="http://schemas.dmtf.org/ovf/environment/1"
2516     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2517     ovfenv:id="DB1">
2518
2519     <!-- Information about hypervisor platform -->
2520     <PlatformSection>
2521         <Kind>ESX Server</Kind>
2522         <Version>3.0.1</Version>
2523         <Vendor>VMware, Inc.</Vendor>
2524         <Locale>en_US</Locale>
2525     </PlatformSection>
2526
2527     <!-- Properties defined for this virtual machine -->
```

```
2528 <PropertySection>
2529   <Property ovfenv:key="com.mydb.db.vm1" ovfenv:value="10.20.132.102"/>
2530   <Property ovfenv:key="com.mydb.db.vm2" ovfenv:value="10.20.132.103"/>
2531   <Property ovfenv:key="com.mydb.db.log" ovfenv:value="warning"/>
2532   <Property ovfenv:key="com.mydb.db.ip" ovfenv:value="10.20.132.102"/>
2533   <Property ovfenv:key="com.mydb.db.ip2" ovfenv:value="10.20.132.103"/>
2534   <Property ovfenv:key="com.mydb.db.primaryAtBoot" ovfenv:value="yes"/>
2535 </PropertySection>
2536
2537 <Entity ovfenv:id="DB2">
2538   <PropertySection>
2539     <Property ovfenv:key="com.mydb.db.vm1" ovfenv:value="10.20.132.102"/>
2540     <Property ovfenv:key="com.mydb.db.vm2" ovfenv:value="10.20.132.103"/>
2541     <Property ovfenv:key="com.mydb.db.log" ovfenv:value="warning"/>
2542     <Property ovfenv:key="com.mydb.db.ip" ovfenv:value="10.20.132.103"/>
2543     <Property ovfenv:key="com.mydb.db.ip2" ovfenv:value="10.20.132.102"/>
2544     <Property ovfenv:key="com.mydb.db.primaryAtBoot" ovfenv:value="no"/>
2545   </PropertySection>
2546 </Entity>
2547 </Environment>
2548
```

2549
2550
2551
2552

ANNEX C (informative)

Single Virtual System LAMP Stack Example

2553 In this example we provide two concrete examples on how an OVF descriptor for a LAMP virtual
2554 appliance could look like. We show both a single-VM LAMP virtual appliance and a multi-VM LAMP virtual
2555 appliance. LAMP is an abbreviation for a service built using the Linux operating system, Apache web
2556 server, MySQL database, and the PHP web development software packages.

2557 This examples show how the *ProductSection* can be used to specify both operating system and
2558 application-level deployment parameters. For example, these parameters can be used to optimize the
2559 performance of a service when deployed into a particular environment. The descriptors are complete, but
2560 otherwise kept minimal, so there are, for example, no EULA sections.

2561 C.1 Deployment-time Customization

2562 A part of the deployment phase of an OVF package is to provide customization parameters. The
2563 customization parameters are specified in the OVF descriptor and are provided to the guest software
2564 using the OVF environment. This deployment time customization is in addition to the virtual machine level
2565 parameters, which includes virtual switch connectivity and physical storage location.

2566 For a LAMP-based virtual appliance, the deployment time customization includes IP address and port
2567 number of the service, network information such as gateway and subnet, and also parameters so the
2568 performance can be optimized for a given deployment. The properties that will be exposed to the
2569 deployer will vary from vendor to vendor and service to service. In our example descriptors, we use the
2570 following set of parameters for the 4 different LAMP components:

2571 **Table 3** illustrates the properties for the LAMP configuration

2572

Table 3 – LAMP Configuration

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

2573 The parameters in *italic* are required configuration from the user. Otherwise, they have reasonable
 2574 defaults, so the user does not necessarily need to provide a value.

2575 The customization parameters for each software product are encapsulated in separate product sections.
 2576 For example, for the Apache web server the following section is used:

```

2577 <ProductSection ovf:class="org.apache.httpd">
2578   <Info>Product customization for the installed Apache Web Server</Info>
2579   <Product>Apache Distribution Y</Product>
2580   <Version>2.6.6</Version>
2581   <Property ovf:key="httpPort" ovf:type="uint16" ovf:value="80"
2582     ovf:userConfigurable="true">
2583     <Description>Port number for HTTP requests</Description>
2584   </Property>
2585   <Property ovf:key="httpsPort" ovf:type="uint16" ovf:value="443"
2586     ovf:userConfigurable="true">
2587     <Description>Port number for HTTPS requests</Description>
2588   </Property>
2589   <Property ovf:key="startThreads" ovf:type="uint16" ovf:value="50"
2590     ovf:userConfigurable="true">
2591     <Description>Number of threads created on startup. </Description>
2592   </Property>
2593   <Property ovf:key="minSpareThreads" ovf:type="uint16" ovf:value="15"
2594     ovf:userConfigurable="true">
2595     <Description>Minimum number of idle threads to handle request
2596 spikes.</Description>
2597   </Property>
2598   <Property ovf:key="maxSpareThreads" ovf:type="uint16" ovf:value="30"
2599     ovf:userConfigurable="true">
2600     <Description>Maximum number of idle threads </Description>
2601   </Property>
2602   <Property ovf:key="maxClients" ovf:type="uint16" ovf:value="256"
2603     ovf:userConfigurable="true">
2604     <Description>Limit the number of simultaneous requests that will be served.
2605 </Description>
2606   </Property>
2607 </ProductSection>
    
```

2608 The `ovf:class="org.apache.httpd"` attribute specifies the prefix for the properties. Hence, the
 2609 Apache database is expected to look for the following properties in the OVF environment:

```

2610 <Environment
2611   ...
2612   <!-- Properties defined for this virtual machine -->
2613   <PropertySection>
2614     <Property ovfenv:name="org.apache.httpd.httpPort ovfenv:value="80"/>
2615     <Property ovfenv:name="org.apache.httpd.httpsPort ovfenv:value="443"/>
2616     <Property ovfenv:name="org.apache.httpd.startThreads" ovfenv:value="50"/>
    
```

```

2617     <Property ovfenv:name="org.apache.httpd.minSpareThreads" ovfenv:value="15"/>
2618     <Property ovfenv:name="org.apache.httpd.maxSpareThreads" ovfenv:value="30"/>
2619     <Property ovfenv:name="org.apache.httpd.maxClients" ovfenv:value="256"/>
2620     ...
2621 </PropertySection>
2622 ...
2623 </Environment>

```

2624 C.2 Simple LAMP OVF Descriptor

2625 A complete OVF descriptor for a single VM virtual appliance with the LAMP stack is listed below:

```

2626 <?xml version="1.0" encoding="UTF-8"?>
2627 <Envelope
2628     xmlns="http://schemas.dmtf.org/ovf/envelope/1"
2629     xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1"
2630     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2631     xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2632 schema/2/CIM_VirtualSystemSettingData"
2633     xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2634 schema/2/CIM_ResourceAllocationSettingData"
2635     <!-- References to all external files -->
2636     <References>
2637         <File ovf:id="lamp" ovf:href="lamp.vmdk" ovf:size="180114671"/>
2638     </References>
2639     <!-- Describes meta-information about all virtual disks in the package. -->
2640     <DiskSection>
2641         <Info>List of the virtual disks used in the package</Info>
2642         <Disk ovf:diskId="lamp" ovf:fileRef="lamp" ovf:capacity="4294967296"
2643             ovf:populatedSize="1924967692"
2644             ovf:format="http://www.vmware.com/specifications/vmdk.html#streamOptimized"/>
2645     </DiskSection>
2646     <!-- Describes all networks used in the package -->
2647     <NetworkSection>
2648         <Info>Logical networks used in the package</Info>
2649         <Network ovf:name="VM Network">
2650             <Description>The network that the LAMP Service will be available
2651             on</Description>
2652         </Network>
2653     </NetworkSection>
2654     <VirtualSystem ovf:id="MyLampService">
2655         <Info>Single-VM Virtual appliance with LAMP stack</Info>
2656         <Name>LAMP Virtual Appliance</Name>
2657         <!-- Overall information about the product -->
2658         <ProductSection>
2659             <Info>Product information for the service</Info>
2660             <Product>Lamp Service</Product>
2661             <Version>1.0</Version>
2662             <FullVersion>1.0.0</FullVersion>
2663         </ProductSection>
2664         <!-- Linux component configuration parameters -->
2665         <ProductSection ovf:class="org.linuxdistx">
2666             <Info>Product customization for the installed Linux system</Info>
2667             <Product>Linux Distribution X</Product>
2668             <Version>2.6.3</Version>
2669             <Property ovf:key="hostname" ovf:type="string">
2670                 <Description>Specifies the hostname for the appliance</Description>
2671             </Property>
2672             <Property ovf:key="ip" ovf:type="string">
2673                 <Description>Specifies the IP address for the appliance</Description>
2674             </Property>

```



```

2675     </Property>
2676     <Property ovf:key="subnet" ovf:type="string">
2677         <Description> Specifies the subnet to use on the deployed network
2678     </Description>
2679     </Property>
2680     <Property ovf:key="gateway" ovf:type="string">
2681         <Description> Specifies the gateway on the deployed network
2682     </Description>
2683     </Property>
2684     <Property ovf:key="dns" ovf:type="string">
2685         <Description> A comma separated list of DNS servers on the deployed
2686         network </Description>
2687     </Property>
2688     <Property ovf:key="netCoreRmemMaxMB" ovf:type="uint16" ovf:value="16"
2689         ovf:userConfigurable="true">
2690         <Description> Specify TCP read max buffer size in mega bytes. Default
2691 is
2692         16. </Description>
2693     </Property>
2694     <Property ovf:key="netCoreWmemMaxMB" ovf:type="uint16" ovf:value="16"
2695         ovf:userConfigurable="true">
2696 is
2697         <Description> Specify TCP write max buffer size in mega bytes. Default
2698         16. </Description>
2699     </Property>
2700 </ProductSection>
2701 <!-- Apache component configuration parameters -->
2702 <ProductSection ovf:class="org.apache.httpd">
2703     <Info>Product customization for the installed Apache Web Server</Info>
2704     <Product>Apache Distribution Y</Product>
2705     <Version>2.6.6</Version>
2706     <Property ovf:key="httpPort" ovf:type="uint16" ovf:value="80"
2707         ovf:userConfigurable="true">
2708         <Description>Port number for HTTP requests</Description>
2709     </Property>
2710     <Property ovf:key="httpsPort" ovf:type="uint16" ovf:value="443"
2711         ovf:userConfigurable="true">
2712         <Description>Port number for HTTPS requests</Description>
2713     </Property>
2714     <Property ovf:key="startThreads" ovf:type="uint16" ovf:value="50"
2715         ovf:userConfigurable="true">
2716         <Description>Number of threads created on startup. </Description>
2717     </Property>
2718     <Property ovf:key="minSpareThreads" ovf:type="uint16" ovf:value="15"
2719         ovf:userConfigurable="true">
2720         <Description> Minimum number of idle threads to handle request spikes.
2721     </Description>
2722     </Property>
2723     <Property ovf:key="maxSpareThreads" ovf:type="uint16" ovf:value="30"
2724         ovf:userConfigurable="true">
2725         <Description>Maximum number of idle threads </Description>
2726     </Property>
2727     <Property ovf:key="maxClients" ovf:type="uint16" ovf:value="256"
2728         ovf:userConfigurable="true">
2729         <Description>Limit the number of simultaneous requests that will be
2730         served. </Description>
2731     </Property>
2732 </ProductSection>
2733 <!-- MySQL component configuration parameters -->
2734 <ProductSection ovf:class="org.mysql.db">
2735     <Info>Product customization for the installed MySql Database Server</Info>
2736     <Product>MySQL Distribution Z</Product>
2737     <Version>5.0</Version>
2738     <Property ovf:key="queryCacheSizeMB" ovf:type="uint16" ovf:value="32"

```

```

2739         ovf:userConfigurable="true">
2740         <Description>Buffer to cache repeated queries for faster access (in
2741         MB)</Description>
2742     </Property>
2743     <Property ovf:key="maxConnections" ovf:type="uint16" ovf:value="500"
2744         ovf:userConfigurable="true">
2745         <Description>The number of concurrent connections that can be
2746         served</Description>
2747     </Property>
2748     <Property ovf:key="waitTimeout" ovf:type="uint16" ovf:value="100"
2749         ovf:userConfigurable="true">
2750         <Description>Number of seconds to wait before timing out a connection
2751         </Description>
2752     </Property>
2753 </ProductSection>
2754 <!-- PHP component configuration parameters -->
2755 <ProductSection ovf:class="net.php">
2756     <Info>Product customization for the installed PHP component</Info>
2757     <Product>PHP Distribution U</Product>
2758     <Version>5.0</Version>
2759     <Property ovf:key="sessionTimeout" ovf:type="uint16" ovf:value="5"
2760         ovf:userConfigurable="true">
2761         <Description> How many minutes a session has to be idle before it is
2762         timed out </Description>
2763     </Property>
2764     <Property ovf:key="concurrentSessions" ovf:type="uint16" ovf:value="500"
2765         ovf:userConfigurable="true">
2766         <Description> The number of concurrent sessions that can be served
2767         </Description>
2768     </Property>
2769     <Property ovf:key="memoryLimit" ovf:type="uint16" ovf:value="32"
2770         ovf:userConfigurable="true">
2771         <Description> How much memory in megabytes a script can consume before
2772         being killed </Description>
2773     </Property>
2774 </ProductSection>
2775 <OperatingSystemSection ovf:id="99">
2776     <Info>Guest Operating System</Info>
2777     <Description>Linux 2.6.x</Description>
2778 </OperatingSystemSection>
2779 <VirtualHardwareSection>
2780     <Info>Virtual Hardware Requirements: 256MB, 1 CPU, 1 disk, 1 NIC</Info>
2781     <System>
2782         <vssd:ElementName>Virtual Hardware Family</vssd:ElementName>
2783         <vssd:InstanceID>0</vssd:InstanceID>
2784         <vssd:VirtualSystemType>vmx-04</vssd:VirtualSystemType>
2785     </System>
2786     <Item>
2787         <rasd:Description>Number of virtual CPUs</rasd:Description>
2788         <rasd:ElementName>1 virtual CPU</rasd:ElementName>
2789         <rasd:InstanceID>1</rasd:InstanceID>
2790         <rasd:ResourceType>3</rasd:ResourceType>
2791         <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2792     </Item>
2793     <Item>
2794         <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
2795         <rasd:Description>Memory Size</rasd:Description>
2796         <rasd:ElementName>256 MB of memory</rasd:ElementName>
2797         <rasd:InstanceID>2</rasd:InstanceID>
2798         <rasd:ResourceType>4</rasd:ResourceType>
2799         <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
2800     </Item>
2801     <Item>
2802         <rasd:AutomaticAllocation>true</rasd:AutomaticAllocation>

```

```
2803         <rasd:Connection>VM Network</rasd:Connection>
2804         <rasd:ElementName>Ethernet adapter on "VM Network"</rasd:ElementName>
2805         <rasd:InstanceID>3</rasd:InstanceID>
2806         <rasd:ResourceType>10</rasd:ResourceType>
2807     </Item>
2808     <Item>
2809         <rasd:ElementName>SCSI Controller 0 - LSI Logic</rasd:ElementName>
2810         <rasd:InstanceID>4</rasd:InstanceID>
2811         <rasd:ResourceSubType>LsiLogic</rasd:ResourceSubType>
2812         <rasd:ResourceType>6</rasd:ResourceType>
2813     </Item>
2814     <Item>
2815         <rasd:ElementName>Harddisk 1</rasd:ElementName>
2816         <rasd:HostResource>ovf:/disk/lamp</rasd:HostResource>
2817         <rasd:InstanceID>5</rasd:InstanceID>
2818         <rasd:Parent>4</rasd:Parent>
2819         <rasd:ResourceType>17</rasd:ResourceType>
2820     </Item>
2821 </VirtualHardwareSection>
2822 </VirtualSystem>
2823 </Envelope>
```

2824

2825
2826
2827
2828

ANNEX D (informative)

Multiple Virtual System LAMP Stack Example

2829 In this example is for an OVF descriptor for a LAMP virtual appliance could look like in a multi-VM LAMP
2830 virtual appliance. LAMP is an abbreviation for a service built using the Linux operating system, Apache
2831 web server, MySQL database, and the PHP web development software packages.

2832 D.1 Two-tier LAMP OVF Descriptor

2833 In a two tier LAMP stack, the application tier (Linux, Apache, PHP) and the database tier (Linux, MySQL)
2834 server) are run as separate virtual machines for greater scalability.

2835 The OVF format makes it largely transparent to the user how a service is implemented. In particular, the
2836 deployment experience when installing a single-VM or a two-tier LAMP appliance is very similar. The only
2837 visible difference is that the user will need to supply two IP addresses and two DNS host names.

2838 As compared to the single-VM descriptor, the following changes are made:

- 2839 • All the user-configurable parameters are put in the *VirtualSystemCollection* entity. The
2840 ProductSections for Apache, MySQL, and PHP are unchanged from the single VM case.
- 2841 • The Linux software in the two virtual machines needs to be configured slightly different (IP and
2842 hostname) while sharing most parameters. A new ProductSection is added to the
2843 *VirtualSystemCollection* to prompt the user, and the `${property}` expression is used to
2844 assign the values in each *VirtualSystem* entity.
- 2845 • Disk chains are used to keep the download size comparable to that of a single VM appliance.
2846 Since the Linux installation is stored on a shared base disk, effectively only one copy of Linux
2847 needs to be downloaded.

2848 The complete OVF descriptor is shown below:

```
2849 <?xml version="1.0" encoding="UTF-8"?>
2850 <Envelope
2851   xmlns="http://schemas.dmtf.org/ovf/envelope/1"
2852   xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1"
2853   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2854   xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2855 schema/2/CIM_VirtualSystemSettingData"
2856   xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2857 schema/2/CIM_ResourceAllocationSettingData"
2858   <!-- References to all external files. -->
2859   <References>
2860     <File ovf:id="lamp-base" ovf:href="lampdb.vmdk" ovf:size="180114671"/>
2861     <File ovf:id="lamp-db" ovf:href="lampdb.vmdk" ovf:size="1801146"/>
2862     <File ovf:id="lamp-app" ovf:href="lampapp.vmdk" ovf:size="34311371"/>
2863   </References>
2864   <!-- Describes meta-information about all virtual disks in the package.
2865     This example is encoded as a delta-disk hierarchy.
2866   -->
2867   <DiskSection>
2868     <Info>List of the virtual disks used in the package</Info>
2869     <Disk ovf:diskId="lamp-base" ovf:fileRef="lamp-base" ovf:capacity="4294967296"
2870       ovf:populatedSize="1924967692"
2871     ovf:format="http://www.vmware.com/specifications/vmdk.html#streamOptimized"/>
2872     <Disk ovf:diskId="lamp-db" ovf:fileRef="lamp-db" ovf:capacity="4294967296"
2873
```

```

2874         ovf:populatedSize="19249672"
2875
2876     ovf:format="http://www.vmware.com/specifications/vmdk.html#streamOptimized"
2877     ovf:parentRef="lamp-base"/>
2878     <Disk ovf:diskId="lamp-app" ovf:fileRef="lamp-app" ovf:capacity="4294967296"
2879     ovf:populatedSize="2349692"
2880
2881     ovf:format="http://www.vmware.com/specifications/vmdk.html#streamOptimized"
2882     ovf:parentRef="lamp-base"/>
2883 </DiskSection>
2884 <!-- Describes all networks used in the package -->
2885 <NetworkSection>
2886     <Info>Logical networks used in the package</Info>
2887     <Network ovf:name="VM Network">
2888         <Description>The network that the LAMP Service will be available
2889         on</Description>
2890     </Network>
2891 </NetworkSection>
2892 <VirtualSystemCollection ovf:id="LampService">
2893     <Info>Virtual appliance with a 2-tier distributed LAMP stack</Info>
2894     <Name>LAMP Service</Name>
2895     <!-- Overall information about the product -->
2896     <ProductSection ovf:class="org.mylamp">
2897         <Info>Product information for the service</Info>
2898         <Product>My Lamp Service</Product>
2899         <Version>1.0</Version>
2900         <FullVersion>1.0.0</FullVersion>
2901     </ProductSection>
2902     <ProductSection ovf:class="org.linuxdist">
2903         <Info>Product customization for Operating System Level</Info>
2904         <Product>Linux Distribution X</Product>
2905         <Version>2.6.3</Version>
2906         <Property ovf:key="dbHostname" ovf:type="string">
2907             <Description>Specifies the hostname for database virtual
2908             machine</Description>
2909         </Property>
2910         <Property ovf:key="appHostname" ovf:type="string">
2911             <Description>Specifies the hostname for application server virtual
2912             machine</Description>
2913         </Property>
2914         <Property ovf:key="dbIp" ovf:type="string">
2915             <Description>Specifies the IP address for the database virtual
2916             machine</Description>
2917         </Property>
2918         <Property ovf:key="appIp" ovf:type="string">
2919             <Description>Specifies the IP address for application server
2920             VM</Description>
2921         </Property>
2922         <Property ovf:key="subnet" ovf:type="string">
2923             <Description> Specifies the subnet to use on the deployed network
2924             </Description>
2925         </Property>
2926         <Property ovf:key="gateway" ovf:type="string">
2927             <Description> Specifies the gateway on the deployed network
2928             </Description>
2929         </Property>
2930         <Property ovf:key="dns" ovf:type="string">
2931             <Description> A comma separated list of DNS servers on the deployed
2932             network </Description>
2933         </Property>
2934         <Property ovf:key="netCoreRmemMaxMB" ovf:type="uint16" ovf:value="16"
2935         ovf:userConfigurable="true">
2936             <Description> Specify TCP read max buffer size in mega bytes. Default
2937
is

```

```
2938         16. </Description>
2939     </Property>
2940     <Property ovf:key="netCoreWmemMaxMB" ovf:type="uint16" ovf:value="16"
2941         ovf:userConfigurable="true">
2942         <Description> Specify TCP write max buffer size in mega bytes. Default
2943 is
2944         16. </Description>
2945     </Property>
2946 </ProductSection>
2947 <!-- Apache component configuration parameters -->
2948 <ProductSection ovf:class="org.apache.httpd">
2949     <Info>Product customization for the installed Apache Web Server</Info>
2950     <Product>Apache Distribution Y</Product>
2951     <Version>2.6.6</Version>
2952     <Property ovf:key="httpPort" ovf:type="uint16" ovf:value="80"
2953         ovf:userConfigurable="true">
2954         <Description>Port number for HTTP requests</Description>
2955     </Property>
2956     <Property ovf:key="httpsPort" ovf:type="uint16" ovf:value="443"
2957         ovf:userConfigurable="true">
2958         <Description>Port number for HTTPS requests</Description>
2959     </Property>
2960     <Property ovf:key="startThreads" ovf:type="uint16" ovf:value="50"
2961         ovf:userConfigurable="true">
2962         <Description>Number of threads created on startup. </Description>
2963     </Property>
2964     <Property ovf:key="minSpareThreads" ovf:type="uint16" ovf:value="15"
2965         ovf:userConfigurable="true">
2966         <Description>Minimum number of idle threads to handle request spikes.
2967         </Description>
2968     </Property>
2969     <Property ovf:key="maxSpareThreads" ovf:type="uint16" ovf:value="30"
2970         ovf:userConfigurable="true">
2971         <Description>Maximum number of idle threads </Description>
2972     </Property>
2973     <Property ovf:key="maxClients" ovf:type="uint16" ovf:value="256"
2974         ovf:userConfigurable="true">
2975         <Description>Limits the number of simultaneous requests that will be
2976         served. </Description>
2977     </Property>
2978 </ProductSection>
2979 <!-- MySQL component configuration parameters -->
2980 <ProductSection ovf:class="org.mysql.db">
2981     <Info>Product customization for the installed MySql Database Server</Info>
2982     <Product>MySQL Distribution Z</Product>
2983     <Version>5.0</Version>
2984     <Property ovf:key="queryCacheSizeMB" ovf:type="uint16" ovf:value="32"
2985         ovf:userConfigurable="true">
2986         <Description>Buffer to cache repeated queries for faster access (in
2987         MB)</Description>
2988     </Property>
2989     <Property ovf:key="maxConnections" ovf:type="uint16" ovf:value="500"
2990         ovf:userConfigurable="true">
2991         <Description>The number of concurrent connections that can be
2992         served</Description>
2993     </Property>
2994     <Property ovf:key="waitTimeout" ovf:type="uint16" ovf:value="100"
2995         ovf:userConfigurable="true">
2996         <Description>Number of seconds to wait before timing out a connection
2997         </Description>
2998     </Property>
2999 </ProductSection>
3000 <!-- PHP component configuration parameters -->
3001 <ProductSection ovf:class="net.php">
```

```

3002     <Info>Product customization for the installed PHP component</Info>
3003     <Product>PHP Distribution U</Product>
3004     <Version>5.0</Version>
3005     <Property ovf:key="sessionTimeout" ovf:type="uint16" ovf:value="5"
3006         ovf:userConfigurable="true">
3007         <Description> How many minutes a session has to be idle before it is
3008             timed out </Description>
3009     </Property>
3010     <Property ovf:key="concurrentSessions" ovf:type="uint16" ovf:value="500"
3011         ovf:userConfigurable="true">
3012         <Description> The number of concurrent sessions that can be served
3013             </Description>
3014     </Property>
3015     <Property ovf:key="memoryLimit" ovf:type="uint16" ovf:value="32"
3016         ovf:userConfigurable="true">
3017         <Description> How much memory in megabytes a script can consume before
3018             being killed </Description>
3019     </Property>
3020 </ProductSection>
3021 <StartupSection>
3022     <Info>Startup order of the virtual machines</Info>
3023     <Item ovf:id="DbServer" ovf:order="1" ovf:startDelay="120"
3024         ovf:startAction="powerOn" ovf:waitingForGuest="true"
3025         ovf:stopDelay="120"
3026         ovf:stopAction="guestShutdown"/>
3027     <Item ovf:id="AppServer" ovf:order="2" ovf:startDelay="120"
3028         ovf:startAction="powerOn" ovf:waitingForGuest="true"
3029         ovf:stopDelay="120"
3030         ovf:stopAction="guestShutdown"/>
3031 </StartupSection>
3032 <VirtualSystem ovf:id="AppServer">
3033     <Info>The configuration of the AppServer virtual machine</Info>
3034     <Name>Application Server</Name>
3035     <!-- Linux component configuration parameters -->
3036     <ProductSection ovf:class="org.linuxdistx">
3037         <Info>Product customization for the installed Linux system</Info>
3038         <Product>Linux Distribution X</Product>
3039         <Version>2.6.3</Version>
3040         <Property ovf:key="hostname" ovf:type="string"
3041             ovf:value="\${appHostName}"/>
3042         <Property ovf:key="ip" ovf:type="string" ovf:value="\${appIp}"/>
3043         <Property ovf:key="subnet" ovf:type="string" ovf:value="\${subnet}"/>
3044         <Property ovf:key="gateway" ovf:type="string" ovf:value="\${gateway}"/>
3045         <Property ovf:key="dns" ovf:type="string" ovf:value="\${dns}"/>
3046         <Property ovf:key="netCoreRmemMaxMB" ovf:type="string"
3047             ovf:value="\${netCoreRmemMaxMB}"/>
3048         <Property ovf:key="netCoreWmemMaxMB" ovf:type="string"
3049             ovf:value="\${netCoreWmemMaxMB}"/>
3050     </ProductSection>
3051     <OperatingSystemSection ovf:id="99">
3052         <Info>Guest Operating System</Info>
3053         <Description>Linux 2.6.x</Description>
3054     </OperatingSystemSection>
3055     <VirtualHardwareSection>
3056         <Info>Virtual Hardware Requirements: 256 MB, 1 CPU, 1 disk, 1
3057         NIC</Info>
3058         <System>
3059             <vssd:ElementName>Virtual Hardware Family</vssd:ElementName>
3060             <vssd:InstanceID>0</vssd:InstanceID>
3061             <vssd:VirtualSystemType>vmx-04</vssd:VirtualSystemType>
3062         </System>
3063         <Item>
3064             <rasd:Description>Number of virtual CPUs</rasd:Description>
3065             <rasd:ElementName>1 virtual CPU</rasd:ElementName>

```

```

3066         <rasd:InstanceID>1</rasd:InstanceID>
3067         <rasd:ResourceType>3</rasd:ResourceType>
3068         <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
3069     </Item>
3070     <Item>
3071         <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
3072         <rasd:Description>Memory Size</rasd:Description>
3073         <rasd:ElementName>256 MB of memory</rasd:ElementName>
3074         <rasd:InstanceID>2</rasd:InstanceID>
3075         <rasd:ResourceType>4</rasd:ResourceType>
3076         <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
3077     </Item>
3078     <Item>
3079         <rasd:AutomaticAllocation>true</rasd:AutomaticAllocation>
3080         <rasd:Connection>VM Network</rasd:Connection>
3081         <rasd:ElementName>Ethernet adapter on "VM
Network"</rasd:ElementName>
3082         <rasd:InstanceID>3</rasd:InstanceID>
3083         <rasd:ResourceSubType>PCNet32</rasd:ResourceSubType>
3084         <rasd:ResourceType>10</rasd:ResourceType>
3085     </Item>
3086     <Item>
3087         <rasd:ElementName>SCSI Controller 0 - LSI Logic</rasd:ElementName>
3088         <rasd:InstanceID>4</rasd:InstanceID>
3089         <rasd:ResourceSubType>LsiLogic</rasd:ResourceSubType>
3090         <rasd:ResourceType>6</rasd:ResourceType>
3091     </Item>
3092     <Item>
3093         <rasd:ElementName>Harddisk 1</rasd:ElementName>
3094         <rasd:HostResource>ovf:/disk/lamp-app</rasd:HostResource>
3095         <rasd:InstanceID>5</rasd:InstanceID>
3096         <rasd:Parent>4</rasd:Parent>
3097         <rasd:ResourceType>17</rasd:ResourceType>
3098     </Item>
3099 </VirtualHardwareSection>
3100 </VirtualSystem>
3101 <VirtualSystem ovf:id="DB Server">
3102     <Info>The configuration of the database virtual machine</Info>
3103     <Name>Database Server</Name>
3104     <!-- Linux component configuration parameters -->
3105     <ProductSection ovf:class="org.linuxdistx">
3106         <Info>Product customization for the installed Linux system</Info>
3107         <Product>Linux Distribution X</Product>
3108         <Version>2.6.3</Version>
3109         <Property ovf:key="hostname" ovf:type="string"
3110             ovf:value="{dbHostName}"/>
3111         <Property ovf:key="ip" ovf:type="string" ovf:value="{dbIp}"/>
3112         <Property ovf:key="subnet" ovf:type="string" ovf:value="{subnet}"/>
3113         <Property ovf:key="gateway" ovf:type="string" ovf:value="{gateway}"/>
3114         <Property ovf:key="dns" ovf:type="string" ovf:value="{dns}"/>
3115         <Property ovf:key="netCoreRmemMaxMB" ovf:type="string"
3116             ovf:value="{netCoreRmemMaxMB}"/>
3117         <Property ovf:key="netCoreWmemMaxMB" ovf:type="string"
3118             ovf:value="{netCoreWmemMaxMB}"/>
3119     </ProductSection>
3120     <OperatingSystemSection ovf:id="99">
3121         <Info>Guest Operating System</Info>
3122         <Description>Linux 2.6.x</Description>
3123     </OperatingSystemSection>
3124     <VirtualHardwareSection>
3125         <Info>Virtual Hardware Requirements: 256 MB, 1 CPU, 1 disk, 1
3126         nic</Info>
3127         <System>
3128             <vssd:ElementName>Virtual Hardware Family</vssd:ElementName>
3129

```



```

3130         <vssd:InstanceID>0</vssd:InstanceID>
3131         <vssd:VirtualSystemType>vmx-04</vssd:VirtualSystemType>
3132     </System>
3133     <Item>
3134         <rasd:Description>Number of virtual CPUs</rasd:Description>
3135         <rasd:ElementName>1 virtual CPU</rasd:ElementName>
3136         <rasd:InstanceID>1</rasd:InstanceID>
3137         <rasd:ResourceType>3</rasd:ResourceType>
3138         <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
3139     </Item>
3140     <Item>
3141         <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
3142         <rasd:Description>Memory Size</rasd:Description>
3143         <rasd:ElementName>256 MB of memory</rasd:ElementName>
3144         <rasd:InstanceID>2</rasd:InstanceID>
3145         <rasd:ResourceType>4</rasd:ResourceType>
3146         <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
3147     </Item>
3148     <Item>
3149         <rasd:AutomaticAllocation>true</rasd:AutomaticAllocation>
3150         <rasd:Connection>VM Network</rasd:Connection>
3151         <rasd:ElementName>Ethernet adapter on "VM
3152 Network"</rasd:ElementName>
3153         <rasd:InstanceID>3</rasd:InstanceID>
3154         <rasd:ResourceType>10</rasd:ResourceType>
3155     </Item>
3156     <Item>
3157         <rasd:ElementName>SCSI Controller 0 - LSI Logic</rasd:ElementName>
3158         <rasd:InstanceID>4</rasd:InstanceID>
3159         <rasd:ResourceSubType>LsiLogic</rasd:ResourceSubType>
3160         <rasd:ResourceType>6</rasd:ResourceType>
3161     </Item>
3162     <Item>
3163         <rasd:ElementName>Harddisk 1</rasd:ElementName>
3164         <rasd:HostResource>ovf:/disk/lamp-db</rasd:HostResource>
3165         <rasd:InstanceID>5</rasd:InstanceID>
3166         <rasd:Parent>4</rasd:Parent>
3167         <rasd:ResourceType>17</rasd:ResourceType>
3168     </Item>
3169 </VirtualHardwareSection>
3170 </VirtualSystem>
3171 </VirtualSystemCollection>
3172 </Envelope>
3173

```

3174
3175
3176

ANNEX E (informative) Extensibility Example

3177 The OVF specification allows custom meta-data to be added to OVF descriptors in several ways:

- 3178 • New section elements can be defined as part of the `Section` substitution group, and used
3179 wherever the OVF schemas allow sections to be present.
- 3180 • The OVF schemas use an open content model, where all existing types can be extended at the
3181 end with additional elements. Extension points are declared in the OVF schemas with `xs:any`
3182 declarations with `namespace="##other"`.
- 3183 • The OVF schemas allow additional attributes on existing types.

3184 Custom meta-data is not allowed to use OVF XML namespaces. On custom elements, a boolean
3185 `ovf:required` attribute specifies whether the information in the element is required for correct behavior
3186 or optional.

3187 The open content model in the OVF schemas only allows extending existing types at the end. Using XML
3188 Schema 1.0 it is not easy to allow for a more flexible open content model, due to the Unique Particle
3189 Attribution rule and the necessity of adding `xs:any` declarations everywhere in the schema. The XML
3190 Schema 1.1 draft standard contains a much more flexible open content mechanism, using
3191 `xs:openContent mode="interleave"` declarations.

3192 E.1 Custom Schema

3193 A custom XML schema defining two extension types is listed below. The first declaration defines a
3194 custom member of the OVF `Section` substitution group, while the second declaration defines a simple
3195 custom type.

```

3196 <?xml version="1.0" encoding="UTF-8"?>
3197 <xs:schema
3198   targetNamespace="http://schemas.customextension.org/1"
3199   xmlns:custom="http://schemas.customextension.org/1"
3200   xmlns="http://schemas.customextension.org/1"
3201   xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1"
3202   xmlns:xs="http://www.w3.org/2001/XMLSchema"
3203   attributeFormDefault="qualified"
3204   elementFormDefault="qualified">
3205
3206   <!-- Define a custom member of the ovf:Section substitution group -->
3207   <xs:element name="CustomSection" type="custom:CustomSection_Type"
3208 substitutionGroup="ovf:Section"/>
3209
3210   <xs:complexType name="CustomSection_Type">
3211     <xs:complexContent>
3212       <xs:extension base="ovf:Section_Type">
3213         <xs:sequence>
3214           <xs:element name="Data" type="xs:string"/>
3215         </xs:sequence>
3216         <xs:anyAttribute namespace="##any" processContents="lax"/>
3217       </xs:extension>
3218     </xs:complexContent>
3219   </xs:complexType>
3220
3221   <!-- Define other simple custom type not part of ovf:Section substitution group --
3222   >
3223   <xs:complexType name="CustomOther_Type">
3224
```

```

3225     <xs:sequence>
3226         <xs:element name="Data" type="xs:string"/>
3227     </xs:sequence>
3228     <xs:attribute ref="ovf:required"/>
3229     <xs:anyAttribute namespace="##any" processContents="lax"/>
3230 </xs:complexType>
3231
3232 </xs:schema >

```

3233 E.2 Descriptor with custom extensions

3234 A complete OVF descriptor using the custom schema above is listed below. The descriptor validates
 3235 against the OVF schema and the custom schema, but apart from extension examples the descriptor is
 3236 kept minimal and is as such not useful.

3237 The descriptor contains all three extension types: a custom OVF `Section` element, a custom element at
 3238 an extension point, and a custom attribute.

```

3239 <?xml version="1.0" encoding="UTF-8"?>
3240 <Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3241     xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-
3242     schema/2/CIM_VirtualSystemSettingData"
3243     xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
3244     schema/2/CIM_ResourceAllocationSettingData"
3245     xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1"
3246     xmlns="http://schemas.dmtf.org/ovf/envelope/1"
3247     xmlns:custom="http://schemas.customextension.org/1">
3248
3249     <!-- Dummy References element -->
3250     <References/>
3251
3252     <!-- EXAMPLE: Optional custom OVF section element with validation against custom
3253     schema -->
3254     <custom:CustomSection ovf:required="false">
3255         <Info>Description of custom extension</Info>
3256         <custom:Data>somevalue</custom:Data>
3257     </custom:CustomSection>
3258
3259     <!-- Describes all networks used in the package -->
3260     <NetworkSection>
3261         <Info>Logical networks used in the package</Info>
3262         <!-- EXAMPLE: Optional custom attribute -->
3263         <Network ovf:name="VM Network" custom:desiredCapacity="1 Gbit/s"/>
3264         <!-- EXAMPLE: Optional custom meta-data inserted at extension point with
3265         validation
3266             against custom schema -->
3267         <custom:CustomOther xsi:type="custom:CustomOther_Type" ovf:required="false">
3268             <custom:Data>somevalue</custom:Data>
3269         </custom:CustomOther>
3270     </NetworkSection>
3271
3272     <!-- Dummy Content element -->
3273     <VirtualSystem ovf:id="Dummy">
3274         <Info>Dummy VirtualSystem</Info>
3275     </VirtualSystem>
3276 </Envelope>

```

3277 The OVF environment XML schemas contain extension mechanisms matching those of the OVF
 3278 envelope XML schemas, so OVF environment documents are similarly extensible.

3279
3280
3281

ANNEX F (informative) Change Log

3282

Version	Date	Description
1.0.0	2009-02-17	First publication
1.0.1	2011-10-20	Errata publication
2.0.0	2013-02-19	wgv 0.2.0 fixed formatting, variables and annex headers, no text changes, baseline for compare
2.0.0	2013-02-20	wgv 0.2.1 added figures 2 & 3, added figure and table labels, inserted Forward
2.0.0	2013-02-21	wgv -0.2.2 updated operational metadata definition and Figure 3, fixed Petstore OVF descriptor
2.0.0	2013-03-26	wgv 0.2.5 updated document structure in clause 3.
2.0.0	2013-03-28	wgv 0.2.6 added in Marvin's edits in clause 2.
2.0.0	2013-05-07	wgv 0.2.7 added Maturi's SharedDiskSection, Shishir's Encryption & Boot Order, Larry extensibility
2.0.0	2013-05-27	wgv 0.2.8 added Marv's input on OperatingSystemSection and EULA section and Peter's input.on authoring clause.
2.0.0	2013-05-30	wgv 0.2.9 Marv's updates on Install and Startup sections
2.0.0	2013-06-19	wgv 0.2.10 Marv's Annotation and Scale Out, Larry's PlacementGroup and Placement
2.0.0	2013-06-19	wgv 0.2.11 Peter's updates
2.0.0	2013-07-01	wgv 0.6.0 updates and clean up in preparation for work in progress
2.0.0a	2013-07-03	Work in progress release

3283