



1
2
3
4

Document Number: DSP0243

Date: 2013-12-12

Version: 2.1.0

5 **Open Virtualization Format Specification**

6 **Document Type: Specification**

7 **Document Status: DMTF Standard**

8 **Document Language: en-US**

9 Copyright notice

10 Copyright © 2010-2013 Distributed Management Task Force, Inc. (DMTF). All rights reserved.

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

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25 withdrawn or modified after publication, and shall be indemnified and held harmless by any party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

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

31 CONTENTS

32 Foreword 6

33 Introduction..... 7

34 1 Scope 9

35 2 Normative references 9

36 3 Terms and definitions 10

37 4 Symbols and abbreviated terms..... 12

38 5 OVF package 13

39 5.1 OVF package structure 13

40 5.2 Virtual disk formats 14

41 5.3 OVF package options 14

42 5.4 Distribution as a set of files 15

43 6 OVF descriptor 15

44 7 Envelope element..... 16

45 7.1 File references 16

46 7.2 Content element..... 17

47 7.3 Extensibility 18

48 7.4 Conformance 18

49 8 Virtual hardware description..... 19

50 8.1 VirtualHardwareSection 19

51 8.2 Extensibility 20

52 8.3 Virtual hardware elements 21

53 8.4 Ranges on elements 22

54 9 Core metadata sections 24

55 9.1 DiskSection 25

56 9.2 NetworkSection 26

57 9.3 ResourceAllocationSection 26

58 9.4 AnnotationSection 27

59 9.5 ProductSection..... 27

60 9.5.1 Property elements 28

61 9.6 EulaSection 30

62 9.7 StartupSection 30

63 9.8 DeploymentOptionSection 31

64 9.9 OperatingSystemSection 32

65 9.10 InstallSection..... 32

66 9.11 EnvironmentFilesSection 33

67 9.12 BootDeviceSection..... 33

68 9.13 SharedDiskSection 34

69 9.14 ScaleOutSection 34

70 9.15 PlacementGroupSection and PlacementSection..... 35

71 9.16 EncryptionSection 37

72 10 Internationalization 38

73 10.1 Internal resource bundles 39

74 10.2 External resource bundles 39

75 10.3 Message content in external file 39

76 11 OVF environment and OVF environment file 39

77 11.1 Transport media 40

78 11.2 Transport media type 41

79 ANNEX A (informative) Symbols and conventions 42

80 ANNEX B (normative) OVF XSD 43

81 ANNEX C (informative) OVF mime type registration template 44

82	ANNEX D (informative) OVF examples	46
83	D.1 Examples of OVF package structure	46
84	D.2 Examples of distribution of files	46
85	D.3 Example of envelope element	47
86	D.4 Example of file references	48
87	D.5 Example of content element	48
88	D.6 Examples of extensibility	48
89	D.7 Examples of VirtualHardwareSection	49
90	D.8 Examples of virtual hardware elements	50
91	D.9 Example of ranges on elements	50
92	D.10 Example of DiskSection	51
93	D.11 Example of NetworkSection	51
94	D.12 Example of ResourceAllocationSection	52
95	D.13 Example of annotation	52
96	D.14 Example of Product section	52
97	D.15 Example of EULA section	53
98	D.16 Example of StartupSection	53
99	D.17 Example of DeploymentOptionSection	53
100	D.18 Example of OperatingSystemSection	54
101	D.19 Example of InstallSection	54
102	D.20 Example of EnvironmentFilesSection	55
103	D.21 Example of BootDeviceSection	55
104	D.22 Example of SharedDiskSection	56
105	D.23 Example of ScaleOutSection	56
106	D.24 Example of PlacementGroupSection	57
107	D.25 Example of EncryptionSection	58
108	D.26 Example of internationalization	59
109	D.27 Example of message content in an external file	60
110	D.28 Example of environment document	61
111	ANNEX E (informative) Network port profile examples	62
112	E.1 Example 1 (OVF descriptor for one virtual system and one network with an inlined network port profile)	62
113	E.2 Example 2 (OVF descriptor for one virtual system and one network with a locally referenced network port profile)	64
114	E.3 Example 3 (OVF descriptor for one virtual system and one network with a network port profile referenced by a URI)	65
115	E.4 Example 4 (OVF descriptor for two virtual systems and one network with two network port profiles referenced by URIs)	67
116	E.5 Example 5 (networkportprofile1.xml)	70
117	E.6 Example 6 (networkportprofile2.xml)	70
118	ANNEX F (informative) Deployment considerations	71
119	F.1 OVF package structure deployment considerations	71
120	F.2 Virtual hardware deployment considerations	71
121	F.3 Core metadata sections deployment considerations	71
122	ANNEX G (informative) Change log	72
123		
124		
125		
126		
127		

128 **Tables**

129 Table 1 – XML namespace prefixes 16

130 Table 2 – Actions for child elements with `ovf:required` attribute 20

131 Table 3 – HostResource element 21

132 Table 4 – Elements for virtual devices and controllers 22

133 Table 5 – Core metadata sections 24

134 Table 6 – Property types 29

135 Table 7 – Property qualifiers 30

136 Table 8 – Availability attributes 36

137 Table 9 – Affinity Attributes 37

138 Table 10 – Allowed combinations of scoped affinity and availability 37

139 Table 11 – Core sections for OEF 40

140

141

Foreword

142 The *Open Virtualization Format Specification* (DSP0243) was prepared by the OVF Work Group of the
143 DMTF.

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

146		
147	Lawrence Lamers	VMware Inc. (Chair & Editor)
148	Hemal Shah	Broadcom Corporation (co-Editor)
149		
150	Hemal Shah	Broadcom Corporation
151	John Crandall	Brocade Communications Systems
152	Marvin Waschke	DMTF Fellow
153	Naveen Joy	Cisco
154	Steven Neely	Cisco
155	Shishir Pardikar	Citrix Systems Inc.
156	Richard Landau	DMTF Fellow
157	Peter Wörndle	Ericsson AB
158	Jacques Durand	Fujitsu
159	Derek Coleman	Hewlett-Packard Company
160	Robert Freund	Hitachi, Ltd.
161	Eric Wells	Hitachi, Ltd.
162	Abdellatif Touimi	Huawei
163	Jeff Wheeler	Huawei
164	Oliver Benke	IBM
165	Ron Doyle	IBM
166	Michael Johanssen	IBM
167	Andreas Maier	IBM
168	John Leung	Intel Corporation
169	Monica Martin	Microsoft Corporation
170	John Parchem	Microsoft Corporation
171	Cheng Wei	Microsoft Corporation
172	Tatyana Bagerman	Oracle
173	Srinivas Maturi	Oracle
174	Dr. Fermín Galán Márquez	Telefónica
175	Miguel Ángel Peñalvo	Telefónica
176	Dr. Fernando de la Iglesia	Telefónica
177	Álvaro Polo	Telefónica
178	Steffen Grarup	VMware Inc.
179	Lawrence Lamers	VMware Inc.
180	Rene Schmidt	VMware Inc.
181	Paul Ferdinand	WBEM Solutions
182	Junsheng Chu	ZTE Corporation
183	Bhumip Khasnabish	ZTE Corporation
184	Ghazanfar Ali	ZTE Corporation

185

Introduction

186 The Open Virtualization Format (OVF) Specification describes an open, secure, efficient and extensible
187 format for the packaging and distribution of software to be run in virtual systems.

188 The OVF package enables the authoring of portable virtual systems and the transport of virtual systems
189 between virtualization platforms. The key properties of the format are as follows:

190 • **Optimized for distribution**

191 OVF supports content verification and integrity checking based on industry-standard public key
192 infrastructure, and it provides a basic scheme for management of software licensing.

193 • **Optimized for a simple, automated user experience**

194 OVF supports validation of the entire package and each virtual system or metadata component
195 of the OVF during the installation phases of the virtual system (VS) lifecycle management
196 process. It also packages with the package relevant user-readable descriptive information that a
197 virtualization platform can use to streamline the installation experience.

198 • **Supports both single VS and multiple-VS configurations**

199 OVF supports both standard single VS packages and packages containing complex, multi-tier
200 services consisting of multiple interdependent VSs.

201 • **Portable VS packaging**

202 OVF is virtualization platform neutral, while also enabling platform-specific enhancements to be
203 captured. It supports the full range of virtual hard disk formats used for hypervisors today, and it
204 is extensible, which allow it to accommodate formats that may arise in the future. Virtual system
205 properties are captured concisely and accurately.

206 • **Vendor and platform independent**

207 OVF does not rely on the use of a specific host platform, virtualization platform, or guest
208 software.

209 • **Extensible**

210 OVF is immediately useful — and extensible. It is designed to be extended as the industry
211 moves forward with virtual appliance technology. It also supports and permits the encoding of
212 vendor-specific metadata to support specific vertical markets.

213 • **Localizable**

214 OVF supports user-visible descriptions in multiple locales, and it supports localization of the
215 interactive processes during installation of an appliance. This capability allows a single
216 packaged appliance to serve multiple market opportunities.

217 • **Open standard**

218 OVF has arisen from the collaboration of key vendors in the industry, and it is developed in an
219 accepted industry forum as a future standard for portable virtual systems.

220 It is not an explicit goal for OVF to be an efficient execution format. A hypervisor is allowed but not
221 required to run software in virtual systems directly out of the Open Virtualization Format.

222

224

Open Virtualization Format Specification

1 Scope

226 The *Open Virtualization Format (OVF) Specification* describes an open, secure, efficient and extensible
227 format for the packaging and distribution of software to be run in virtual systems.

228 The OVF package enables the authoring of portable virtual systems and the transport of virtual systems
229 between virtualization platforms. This version of the specification (2.1) is intended to allow OVF 1.x tools
230 to work with OVF 2.x descriptors in the following sense:

- 231 • Existing OVF 1.x tools should be able to parse OVF 2.x descriptors.
- 232 • Existing OVF 1.x tools should be able to give warnings/errors if dependencies to 2.x features
233 are required for correct operation.

234 If a conflict arises between the schema, text, or tables, the order of precedence to resolve the conflicts is
235 schema; then text; then tables. Figures are for illustrative purposes only and are not a normative part of
236 the standard.

237 A table may constrain the text but it shall not conflict with it.

238 The profile conforms to the cited CIM Schema classes where used. Any requirements contained in the
239 cited CIM Schema classes shall be met. If a conflict arises the CIM Schema takes precedence.

240 The profile conforms to the cited OVF XML Schema. It may constrain the schema but it shall not conflict
241 with it. If a conflict arises the OVF XML Schema takes precedence.

2 Normative references

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

247 DMTF DSP0004, *Common Information Model (CIM) Infrastructure Specification 2.7*,
248 http://www.dmtf.org/standards/published_documents/DSP0004_2.7.pdf

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

251 DMTF DSP0230, *WS-CIM Mapping Specification 1.0*,
252 http://www.dmtf.org/sites/default/files/standards/documents/DSP0230_1.0.2.pdf

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

255 DMTF DSP1041, *Resource Allocation Profile (RAP) 1.1*,
256 http://www.dmtf.org/standards/published_documents/DSP1041_1.1.pdf

257 DMTF DSP1043, *Allocation Capabilities Profile (ACP) 1.0*,
258 http://www.dmtf.org/standards/published_documents/DSP1043_1.0.pdf

259 DMTF DSP1047, *Storage Resource Virtualization Profile 1.0*,
260 http://www.dmtf.org/standards/published_documents/DSP1047_1.0.pdf

- 261 DMTF DSP1050, Ethernet Port Resource Virtualization Profile 1.0,
262 http://www.dmtf.org/standards/published_documents/DSP1050_1.0.pdf
- 263 DMTF DSP1057, *Virtual System Profile 1.0*,
264 http://www.dmtf.org/standards/published_documents/DSP1057_1.0.pdf
- 265 DMTF DSP8023, *OVF XML Schema Specification for OVF Envelope 2.0*,
266 http://schemas.dmtf.org/ovf/envelope/2/dsp8023_2.0.xsd
- 267 DMTF DSP8027, *OVF XML Schema Specification for OVF Environment 1.1*,
268 http://schemas.dmtf.org/ovf/environment/1/dsp8027_1.1.xsd
- 269 DMTF DSP8049, *Network Port Profile XML Schema*,
270 http://schemas.dmtf.org/ovf/networkportprofile/1/dsp8049_1.0.xsd
- 271 IETF RFC1738, T. Berners-Lee, *Uniform Resource Locators (URL)*, December 1994,
272 <http://tools.ietf.org/html/rfc1738>
- 273 IETF RFC1952, P. Deutsch, *GZIP file format specification version 4.3*, May 1996,
274 <http://tools.ietf.org/html/rfc1952>
- 275 IETF RFC2616, R. Fielding et al, *Hypertext Transfer Protocol – HTTP/1.1*, June 1999,
276 <http://tools.ietf.org/html/rfc2616>
- 277 IETF Standard 66, *Uniform Resource Identifiers (URI): Generic Syntax*,
278 <http://tools.ietf.org/html/rfc3986>
- 279 IETF Standard 68, *Augmented BNF for Syntax Specifications: ABNF*,
280 <http://tools.ietf.org/html/rfc5234>
- 281 ISO 9660, 1988 Information processing-Volume and file structure of CD-ROM for information interchange,
282 http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=17505
- 283 ISO, ISO/IEC Directives, Part 2, *Rules for the structure and drafting of International Standards*,
284 <http://isotc.iso.org/livelink/livelink.exe?func=ll&objId=4230456&objAction=browse&sort=subtype>
- 285 [ISO/IEC/IEEE 9945:2009](http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50516): Information technology -- Portable Operating System Interface (POSIX®) Base
286 Specifications, Issue 7
287 http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50516
- 288 W3C, *XML Schema Part 1: Structures Second Edition*. 28 October 2004. W3C Recommendation. URL:
289 <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/>
- 290 W3C, *XML Schema Part 2: Datatypes Second Edition*. 28 October 2004. W3C Recommendation. URL:
291 <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/>
- 292 W3C, XML Encryption Syntax and Processing Version 1.1, 13 March 2012, W3C Candidate
293 Recommendation
294 <http://www.w3.org/TR/2012/CR-xmlenc-core1-20120313/>
- 295 FIPS 180-2: Secure Hash Standard (SHS)
296 <http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf>

297 3 Terms and definitions

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

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

306 The terms "clause", "subclause", "paragraph", and "annex" in this document are to be interpreted as
307 described in [ISO/IEC Directives, Part 2](#), Clause 5.

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

311 The terms defined in [DSP0004](#), [DSP0223](#), and [DSP1001](#) apply to this document. The following additional
312 terms are used in this document.

313 **3.1**

314 **authoring function**

315 the creation of the OVF package

316 **3.2**

317 **chassis**

318 a placement policy as defined in the class CIM_Chassis

319 **3.3**

320 **conditional**

321 indicates requirements to be followed strictly to conform to the document when the specified conditions
322 are met

323 **3.4**

324 **deployment function**

325 a function the result of which is a prepared virtual system

326 **3.5**

327 **geographic**

328 a placement policy referring to a geographic location (e.g., a country, a state, a province, a latlong)

329 **3.6**

330 **guest software**

331 the software that runs inside a virtual system

332 **3.7**

333 **mandatory**

334 indicates requirements to be followed strictly to conform to the document and from which no deviation is
335 permitted

336 **3.8**

337 **optional**

338 indicates a course of action permissible within the limits of the document

339 **3.9**

340 **rack**

341 a placement policy as defined in the class CIM_Rack

- 342 **3.10**
343 **site**
344 a placement policy as defined in Access, Terminals, Transmission and Multiplexing (ATTM); Broadband
345 Deployment - Energy Efficiency and Key Performance Indicators; Part 2: Network sites; Sub-part 1:
346 Operator sites, Technical Report, ETSI TR 105 174-2-1 V1.1.1 (2009-10)
- 347 **3.11**
348 **OVF package**
349 a single compressed file or a set of files that contains the OVF descriptor file and may contain associated
350 virtual disks, operational metadata, and other files
- 351 **3.12**
352 **OVF descriptor**
353 an XML file that validates to [DSP8023](#) and provides the information needed to deploy the OVF package
- 354 **3.13**
355 **virtualization platform**
356 the hypervisor on which the virtual systems run
- 357 **3.14**
358 **virtual appliance**
359 a service delivered as a software stack that utilizes one or more virtual systems
- 360 **3.15**
361 **virtual hardware**
362 the processor, memory and I/O resources provided by a virtualization platform that supports a virtual
363 system
- 364 **3.16**
365 **virtual system**
366 as defined in the Virtual System Profile plus the guest software if any
- 367 **3.17**
368 **virtual system collection**
369 a collection of virtual systems
- 370 **3.18**
371 **virtualization management**
372 the software that performs resource allocation and management of virtual systems

373 **4 Symbols and abbreviated terms**

374 The abbreviations defined in [DSP0004](#), [DSP0223](#), and [DSP1001](#) apply to this document. The following
375 additional abbreviations are used in this document.

- 376 **4.1**
377 **CIM**
378 Common Information Model
- 379 **4.2**
380 **IP**
381 Internet Protocol

382 **4.3**
 383 **OVF**
 384 Open Virtualization Format
 385 **4.4**
 386 **VS**
 387 virtual system
 388 **4.5**
 389 **VSC**
 390 virtual system collection
 391

392 **5 OVF package**

393 **5.1 OVF package structure**

394 An OVF package shall consist of the following files:

- 395 • one OVF descriptor with extension `.ovf`
- 396 • zero or one OVF manifest with extension `.mf`
- 397 • zero or one OVF certificate with extension `.cert`
- 398 • zero or more disk image files
- 399 • zero or more additional resource files, such as ISO images

400 The file extensions `.ovf`, `.mf` and `.cert` shall be used. See D.1 for an example.

401 An OVF package can be stored as either a single compressed file (`.ova`) or a set of files, as described in
 402 5.3 and 5.4. Both modes shall be supported.

403 An OVF package may have a manifest file containing the SHA digests of individual files in the package.
 404 OVF packages authored according to this version of the specification shall use SHA256 digests. The
 405 manifest file shall have an extension `.mf` and the same base name as the `.ovf` file and be a sibling of the
 406 `.ovf` file. If the manifest file is present, a consumer of the OVF package should verify the digests in the
 407 manifest file in the OVF package by computing the actual SHA digests and comparing them with the
 408 digests listed in the manifest file. The manifest file shall contain SHA digests for all distinct files
 409 referenced in the `References` element of the OVF descriptor and for no other files. See clause 7.1

410 The syntax definitions below use ABNF with the exceptions listed in ANNEX A.

411 The format of the manifest file is as follows:

```

412 manifest_file = *( file_digest )
413 file_digest  = algorithm "(" file_name ")" "=" sp digest nl
414 algorithm    = "SHA1" | "SHA256"
415 digest       = *( hex-digit )
416 hex-digit    = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" | "a" |
417 "b" | "c" | "d" | "e" | "f"
418 sp           = %x20
419 nl           = %x0A
  
```

420 See D.1 for an example.

421 An OVF package may be signed by signing the manifest file. The digest of the manifest file is stored in a
 422 certificate file with extension `.cert` file along with the base64-encoded X.509 certificate. The `.cert` file
 423 shall have the same base name as the `.ovf` file and be a sibling of the `.ovf` file.

424 See ANNEX F for deployment considerations.

425 The format of the certificate file shall be as follows:

```

426 certificate_file = manifest_digest certificate_part
427 manifest_digest = algorithm "(" file_name ")" "=" sp signed_digest nl
428 algorithm       = "SHA1" | "SHA256"
429 signed_digest   = *( hex-digit)
430 certificate_part = certificate_header certificate_body certificate_footer
431 certificate_header = "-----BEGIN CERTIFICATE-----" nl
432 certificate_footer = "-----END CERTIFICATE-----" nl
433 certificate_body   = base64-encoded-certificate nl
434                   ; base64-encoded-certificate is a base64-encoded X.509
435                   ; certificate, which may be split across multiple lines
436 hex-digit        = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" | "a"
437 | "b" | "c" | "d" | "e" | "f"
438 sp               = %x20
439 nl               = %x0A
  
```

440 See D.1 for an example.

441 The manifest and certificate files, when present, shall not be included in the `References` section of the
 442 OVF descriptor (see 7.1). This ensures that the OVF descriptor content does not depend on whether the
 443 OVF package has a manifest or is signed, and the decision to add a manifest or certificate to a package
 444 can be deferred to a later stage.

445 The file extensions `.mf` and `.cert` may be used for other files in an OVF package, as long as they do not
 446 occupy the sibling URLs or path names where they would be interpreted as the package manifest or
 447 certificate.

448 5.2 Virtual disk formats

449 OVF does not require any specific disk format to be used, but to comply with this specification the disk
 450 format shall be given by a URI that identifies an unencumbered specification on how to interpret the disk
 451 format. The specification need not be machine readable, but it shall be static and unique so that the URI
 452 may be used as a key by software reading an OVF package to uniquely determine the format of the disk.
 453 The specification shall provide sufficient information so that a skilled person can properly interpret the
 454 disk format for both reading and writing of disk data. The URI should be resolvable.

455 5.3 OVF package options

456 An OVF package may be stored as a compressed OVF package or as a set of files in a directory
 457 structure. A compressed OVF package is stored as single file. The file extension is `.ova` (open virtual
 458 appliance or application). See D.2 for an example.

459 All file references in the OVF descriptor are relative-path references and are described in section 7.1.
 460 Entries in a compressed OVF package shall exist only once.

461 In addition, the entries shall be in one of the following orders inside the OVF package:

462 1) OVF descriptor
463 2) The remaining files shall be in the same order as listed in the References section (see
464 7.1). Note that any external string resource bundle files for internationalization shall be
465 first in the References section (see clause 10).

466 or

467 1) OVF descriptor
468 2) OVF manifest
469 3) OVF certificate
470 4) The remaining files shall be in the same order as listed in the References section (see
471 7.1). Note that any external string resource bundle files for internationalization shall be
472 first in the References section (see clause 10).

473 or

474 1) OVF descriptor
475 2) The intermediate files shall be in the same order as listed in the References section (see
476 7.1). Note that any external string resource bundle files for internationalization shall be
477 first in the References section (see clause 10).
478 3) OVF manifest
479 4) OVF certificate

480 The ordering restriction ensures that it is possible to extract the OVF descriptor from a compressed OVF
481 package without scanning the entire archive. The ordering restriction enables the efficient generation of a
482 compressed OVF package-

483 A compressed OVF package shall be created by using the TAR format that complies with the USTAR
484 (Uniform Standard Tape Archive) format as defined by the [ISO/IEC/IEEE 9945:2009](#).

485 5.4 Distribution as a set of files

486 An OVF package may be made available as a set of files. See D.2 for an example.

487 6 OVF descriptor

488 The OVF descriptor contains the metadata about the OVF package. This is an extensible XML document
489 for encoding information, such as product details, virtual hardware requirements, and licensing.

490 [DSP8023](#) is the schema definition file for the OVF descriptor that contains the elements and attributes.
491 The OVF descriptor shall validate against [DSP8023](#).

492 Clauses 7, 8, and 9, describe the semantics, structure, and extensibility framework of the OVF descriptor.
493 These clauses are not a replacement for reading the schema definitions, but they complement the
494 schema definitions.

495 The XML namespaces used in this specification are listed in Table 1. The choice of any namespace prefix
496 is arbitrary and not semantically significant.

497

Table 1 – XML namespace prefixes

Prefix	XML Namespace
ovf	http://schemas.dmtf.org/ovf/envelope/2
ovfenv	http://schemas.dmtf.org/ovf/environment/1
rasd	http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/ CIM_ResourceAllocationSettingData.xsd
vssd	http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/ CIM_VirtualSystemSettingData.xsd
epasd	http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/ CIM_EthernetPortAllocationSettingData.xsd
sasd	http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/ CIM_StorageAllocationSettingData.xsd
cim	http://schemas.dmtf.org/wbem/wscim/1/common.xsd

498 7 Envelope element

499 The `Envelope` element describes all metadata for the virtual systems (including virtual hardware), as well
500 as the structure of the OVF package itself.

501 The outermost level of the envelope consists of the following parts:

- 502 • A version indication, defined by the XML namespace URIs
- 503 • A list of file references to all external files that are part of the OVF package, defined by the
504 `References` element and its `File` child elements, e.g., virtual disk files, ISO images, and
505 internationalization resources
- 506 • A metadata part, defined by section elements, defined in clause 9
- 507 • A description of the content, either a single virtual system (`VirtualSystem` element) or a
508 collection of multiple virtual systems (`VirtualSystemCollection` element)
- 509 • A specification of message resource bundles for zero or more locales, defined by a `Strings`
510 element for each locale

511 See D.3 for an example.

512 The `xml:lang` attribute on the `Envelope` element is optional. If present, it shall specify the default locale
513 for messages in the descriptor. The `Strings` element is optional. If present, it shall contain string
514 resource bundles for different locales. See clause 10 for more details about internationalization support.

515 7.1 File references

516 The file reference part defined by the `References` element allows a tool to determine the integrity of an
517 OVF package without having to parse or interpret the entire structure of the descriptor. Tools can safely
518 manipulate (for example, copy or archive) OVF packages with no risk of losing files.

519 External string resource bundle files for internationalization shall be placed first in the `References`
520 element. See clause 10 for details.

521 Each `File` element in the reference part shall be given an identifier using the `ovf:id` attribute. The
522 identifier shall be unique inside an OVF package. Each `File` element shall be specified using the
523 `ovf:href` attribute, which shall contain a URL. Relative-path references and the URL schemes "file",

524 "http", and "https" shall be supported, (see [RFC1738](#) and [RFC3986](#)). Relative path references shall
 525 not contain "." dot-segments. (add this to 7.1). Other URL schemes should not be used. If no URL
 526 scheme is specified, the value of the `ovf:href` attribute shall be interpreted as a path name of the
 527 referenced file relative to the location of the OVF descriptor itself. The relative path name shall use the
 528 syntax of relative-path references in [RFC3986](#). The referenced file shall exist. Two different `File`
 529 elements shall not reference the same file with their `ovf:href` attributes.

530 The size of the referenced file may be specified using the `ovf:size` attribute. The unit of this attribute
 531 shall be bytes. If present, the value of the `ovf:size` attribute should match the actual size of the
 532 referenced file.

533 Each file referenced by a `File` element may be compressed using gzip (see [RFC1952](#)). When a `File`
 534 element is compressed using gzip, the `ovf:compression` attribute shall be set to "gzip". Otherwise, the
 535 `ovf:compression` attribute shall be set to "identity" or the entire attribute omitted. Alternatively, if the
 536 href is an HTTP or HTTPS URL, the compression may be specified by the HTTP server by using the
 537 HTTP header `Content-Encoding: gzip` (see [RFC2616](#)). Using HTTP content encoding in combination
 538 with the `ovf:compression` attribute is allowed, but in general does not improve the compression ratio.
 539 When compression is used, the `ovf:size` attribute shall specify the size of the actual compressed file.

540 Files referenced from the reference part may be split into chunks to accommodate file size restrictions on
 541 certain file systems. Chunking shall be indicated by the presence of the `ovf:chunkSize` attribute; the
 542 value of `ovf:chunkSize` attribute shall be the size of each chunk, except the last chunk, which may be
 543 smaller.

544 If the `ovf:chunkSize` attribute is specified, the `File` element shall reference a chunk file representing a
 545 chunk of the entire file. In this case, the value of the `ovf:href` attribute specifies only a part of the URL,
 546 and the syntax for the URL resolving to the chunk file shall be as follows:

```
547 chunk-url      = href-value "." chunk-number
548 chunk-number  = 9(decimal-digit)
549 decimal-digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
```

550 The syntax is defined in ABNF notation with the exceptions listed in ANNEX A. The href-value shall be
 551 the value of the `ovf:href` attribute. The chunk-number shall be the 0-based position of the chunk starting
 552 with the value 0 and increasing with increments of 1 for each chunk.

553 If chunking is combined with compression, the entire file shall be compressed before chunking and each
 554 chunk shall be an equal slice of the compressed file, except for the last chunk which may be smaller.

555 If the OVF package has a manifest file, the file name in the manifest entries shall match the value of the
 556 `ovf:href` attribute for the file, except if the file is split into multiple chunks, in which case the `chunk-url`
 557 shall be used, and the manifest file shall contain an entry for each individual chunk. If chunked files are
 558 used, the manifest file may contain an entry for the entire file; and if present, this digest shall also be
 559 verified. See D.4 for an example.

560 7.2 Content element

561 Virtual system configurations in an OVF package are represented by a `VirtualSystem` or
 562 `VirtualSystemCollection` element. These elements shall be given an identifier using the `ovf:id`
 563 attribute. Direct child elements of a `VirtualSystemCollection` shall have unique identifiers.

564 In the OVF Schema, the `VirtualSystem` and `VirtualSystemCollection` elements are part of a
 565 substitution group with the `Content` element as head of the substitution group. The `Content` element is
 566 abstract and cannot be used directly. The OVF descriptor shall have one or more `Content` elements.

567 The `VirtualSystem` element describes a single virtual system and is a container of section elements.
568 These section elements describe virtual hardware, resources, and product information as defined in
569 clauses 8 and 9. See D.5 for an example.

570 The `VirtualSystemCollection` element is a container of zero or more `VirtualSystem` or
571 `VirtualSystemCollection` elements. Thus, arbitrary complex configurations can be described. The
572 section elements at the `VirtualSystemCollection` level describe appliance information, properties, and
573 resource requirements as defined in clause 9. See D.5 for an example.

574 All elements in the `Content` substitution group shall contain an `Info` element and may contain a `Name`
575 element. The `Info` element contains a human readable description of the meaning of this entity. The `Name`
576 element is a localizable display name of the content. Clause 10 defines how to localize the `Info` and `Name`
577 element.

578 7.3 Extensibility

579 Custom metadata may be added to OVF descriptors in several ways:

- 580 • New section elements may be defined as part of the `Section` substitution group, and used
581 where the OVF Schemas allow sections to be present. All subtypes of the `Section` element
582 shall contain an `Info` element that contains a human-readable description of the meaning of
583 this entity. The values of `Info` elements can be used, for example, to give meaningful warnings
584 to users when a section is being skipped, even if the parser does not know anything about the
585 section. Clause 10 defines how to localize the `Info` element.
- 586 • The OVF Schemas use an open content model, where all existing types may be extended at the
587 end with additional elements. Extension points are declared in the OVF Schemas with `xs:any`
588 declarations with `namespace="##other"`.
- 589 • The OVF Schemas allow additional attributes on existing types.

590 Custom extensions shall not use XML namespaces defined in this specification. This applies to both
591 custom elements and custom attributes.

592 If custom elements are used, the `ovf:required` attribute specifies whether the information in the element
593 is mandatory or is optional. If not specified, the `ovf:required` attribute defaults to TRUE, i.e., mandatory.
594 A deployment function that detects a custom element that is mandatory and that it does not understand
595 shall fail.

596 If custom attributes are used, the information contained in them shall not be required for correct behavior.

597 If a `Section` element defined in the OVF Schema is used and it contains additional child elements that
598 are not understood and the value of their `ovf:required` attribute is TRUE, the deployment function shall
599 fail.

600 See D.6 for an example.

601 7.4 Conformance

602 This standard defines three conformance levels for OVF descriptors, with 1 being the highest level of
603 conformance:

- 604 • Conformance Level: 1 - The OVF descriptor uses only sections and elements and attributes that
605 are defined in this specification.
- 606 • Conformance Level: 2 - The OVF descriptor uses custom sections or elements or attributes that
607 are not defined in this specification and all such extensions are optional as defined in 7.3.

608 Conformance Level: 3 - The OVF descriptor uses custom sections or elements that are not
609 defined in this specification and at least one such extension is required as defined in 7.3. The
610 definition of all required extensions shall be publicly available in an open and unencumbered XML
611 Schema. The complete specification may be inclusive in the XML Schema or available as a
612 separate document.

613 The use of conformance level 3 should be avoided if the OVF package is intended to be portable.

614 The conformance level is not specified directly in the OVF descriptor but shall be determined by the
615 above rules.

616 8 Virtual hardware description

617 8.1 VirtualHardwareSection

618 The `VirtualHardwareSection` element can be used to describe the virtual hardware used by the virtual
619 system.

620 This standard allows incomplete virtual hardware descriptions.

621 The virtualization platform may create additional virtual hardware devices.

622 The virtual hardware devices listed in the `VirtualHardwareSection` element shall be realized.

623

624 This virtual hardware description is based on the CIM classes `CIM_VirtualSystemSettingData`,
625 `CIM_ResourceAllocationSettingData`, `CIM_EthernetPortAllocationSettingData`, and
626 `CIM_StorageAllocationSettingData`. The XML representation of the CIM model is based on the WS-
627 CIM mapping as defined in [DSP0230](#).

628 NOTE This means that the XML elements that belong to the class complex type should be ordered by Unicode
629 code point (binary) order of their CIM property name identifiers. See D.7 for an example.

630 A `VirtualSystem` element shall have a `VirtualHardwareSection` direct child element. The
631 `VirtualHardwareSection` shall not be a direct child element of a `VirtualSystemCollection` element or
632 of an `Envelope` element.

633 One or more `VirtualHardwareSection` elements may occur within a `VirtualSystem` element. See
634 ANNEX F for virtual hardware deployment considerations. If more than one `VirtualHardwareSection`
635 element occurs, an `ovf:id` attribute shall be used to identify the element. If present, the `ovf:id` attribute
636 value shall be unique within the `VirtualSystem` element.

637 The `ovf:transport` attribute specifies the transport media type by which `property` elements are passed
638 to the virtual system. See 9.5 for a description of `property` elements. See 11.2 for a description of
639 transport types.

640 A `VirtualHardwareSection` element contains child elements that describe virtual system and virtual
641 hardware resources (CPU, memory, network, and storage).

642 A `VirtualHardwareSection` element shall have the following direct child elements:

- 643 • zero or one `System` elements
- 644 • zero or more `Item` elements
- 645 • zero or more `EthernetPortItem` elements
- 646 • zero or more `StorageItem` elements.

647 The `System` element is an XML representation of the values of one or more properties of the CIM class
 648 `CIM_VirtualSystemSettingData`. The `vssd:VirtualSystemType`, a direct child element of `System`
 649 element, specifies a virtual system type identifier, which is an implementation defined string that uniquely
 650 identifies the type of the virtual system. Zero or more virtual system type identifiers may be specified,
 651 separated by single space character. In order for the OVF virtual system to be deployable on a target
 652 platform, the virtual system on the target platform should support at least one of the virtual system types
 653 identified in the `vssd:VirtualSystemType` elements. The virtual system type identifiers specified in
 654 `vssd:VirtualSystemType` elements are expected to be matched against the values of property
 655 `VirtualSystemTypesSupported` of CIM class `CIM_VirtualSystemManagementCapabilities`.

656 The virtual hardware characteristics are described as a sequence of `Item` elements. The `Item` element
 657 is an XML representation of an instance of the CIM class `CIM_ResourceAllocationSettingData`. The
 658 element can describe all memory and CPU requirements as well as virtual hardware devices.

659 Multiple device subtypes may be specified in an `Item` element, separated by a single space (0x20)
 660 character.

661 The network hardware characteristics are described as a sequence of `EthernetPortItem` elements. The
 662 `EthernetPortItem` element is an XML representation of the values of one or more properties of the CIM
 663 class `CIM_EthernetPortAllocationSettingData`.

664 The storage hardware characteristics are described as a sequence of `StorageItem` elements. The
 665 `StorageItem` element is an XML representation of the values of one or more properties of the CIM class
 666 `CIM_StorageAllocationSettingData`.

667 8.2 Extensibility

668 The `ovf:required` attribute is optional on the `Item`, `EthernetPortItem`, or `StorageItem` elements. If
 669 used it specifies whether the realization of the element is required for correct behavior of the guest
 670 software. If not specified, `ovf:required` defaults to TRUE.

671 On child elements of the `Item`, `EthernetPortItem`, or `StorageItem` elements, the `ovf:required`
 672 attribute shall be interpreted, even though these elements are in a different RASD WS-CIM namespace.
 673 A tool parsing an `Item` element should act according to Table 2.

674 **Table 2 – Actions for child elements with `ovf:required` attribute**

Child Element	<code>ovf:required</code> Attribute Value	Action
Known	TRUE or not specified	Shall interpret <code>Item</code> , <code>EthernetPortItem</code> , or <code>StorageItem</code>
Known	FALSE	Shall interpret <code>Item</code> , <code>EthernetPortItem</code> , or <code>StorageItem</code>
Unknown	TRUE or not specified	Shall fail <code>Item</code> , <code>EthernetPortItem</code> , or <code>StorageItem</code>
Unknown	FALSE	Shall ignore Child element

675 **8.3 Virtual hardware elements**

676 The element type of the `Item` element in a `VirtualHardwareSection` element is
 677 `CIM_ResourceAllocationSettingData_Type` as defined in `CIM_ResourceAllocationSettingData`.
 678 See ANNEX B.

679 The child elements of `Item` represent the values of one or more properties exposed by the
 680 `CIM_ResourceAllocationSettingData` class. They have the semantics of defined settings as defined in
 681 [DSP1041](#), any profiles derived from [DSP1041](#) for specific resource types, and this standard. See D.8 for
 682 an example.

683 The element type of the `EthernetPortItem` element in a `VirtualHardwareSection` element is
 684 `CIM_EthernetPortAllocationSettingData_Type` as defined in
 685 `CIM_EthernetPortAllocationSettingData`. See ANNEX B.

686 The child elements represent the values of one or more properties exposed by the
 687 `CIM_EthernetPortAllocationSettingData` class. They have the semantics of defined resource
 688 allocation setting data as defined in [DSP1050](#), any profiles derived from [DSP1050](#) for specific Ethernet
 689 port resource types, and this standard. See D.8 for an example.

690 The element type of the `StorageItem` element in a `VirtualHardwareSection` element is
 691 `CIM_StorageAllocationSettingData_Type` as defined in `CIM_StorageAllocationSettingData`. See
 692 ANNEX B

693 The child elements represent the values of one or more properties exposed by the
 694 `CIM_StorageAllocationSettingData` class. They have the semantics of defined resource allocation
 695 setting data as defined in [DSP1047](#), any profiles derived from [DSP1047](#) for specific storage resource
 696 types, and this standard. See D.8 for an example.

697 The `Description` element is used to provide additional metadata about the `Item`, `EthernetPortItem`, or
 698 `StorageItem` element itself. This element enables a consumer of the OVF package to provide descriptive
 699 information about all items, including items that were unknown at the time the application was written.

700 The `Caption`, `Description` and `ElementName` elements are localizable using the `ovf:msgid` attribute
 701 from the OVF envelope namespace. See clause 10 for more details about internationalization support.

702 The optional `ovf:configuration` attribute contains a list of configuration names. See 9.8 on deployment
 703 options for semantics of this attribute. The optional `ovf:bound` attribute is used to specify ranges; see 8.4.

704 All Ethernet adapters in the OVF package that connect to the same network shall have a `Connection`
 705 element that contains the same logical network name. If a `Connection` element is used to represent a
 706 network, the corresponding network shall be represented as a child element of the `NetworkSection`
 707 element with a `name` attribute that matches the value of the `Connection` element.

708 The `HostResource` element is used to refer to resources included in the OVF descriptor as well as logical
 709 devices on the deployment function. Values for `HostResource` elements referring to resources included in
 710 the OVF descriptor are formatted as URIs as specified in Table 3.

711 **Table 3 – HostResource element**

Content	Description
<code>ovf:/file/<id></code>	A reference to a file in the OVF, as specified in the References section. <code><id></code> shall be the value of the <code>ovf:id</code> attribute of the <code>File</code> element being referenced.
<code>ovf:/disk/<id></code>	A reference to a virtual disk, as specified in the <code>DiskSection</code> or <code>SharedDiskSection</code> . <code><id></code> shall be the value of the <code>ovf:diskId</code> attribute of the <code>Disk</code> element being referenced.

712 See ANNEX F for virtual hardware deployment considerations. More than one backing for a device shall
713 not be specified in a `VirtualHardware` element.

714 Table 4 gives a brief overview on how elements from RASD, EPASD, and SASD namespaces are used
715 to describe virtual devices and controllers.

716 **Table 4 – Elements for virtual devices and controllers**

Element	Usage
Description	Is a human-readable description of the meaning of the information. For example, "Specifies the memory size of the virtual system".
ElementName	Is a human-readable description of the content.
InstanceID	Specifies a unique instance ID of the element within the section.
HostResource	Specifies how a virtual device connects to a resource on the virtualization platform. Not all devices need a backing. See Table 3.
ResourceType OtherResourceType ResourceSubtype	Specifies the kind of device that is being described.
AutomaticAllocation	For devices that are connectable, such as floppies, CD-ROMs, and Ethernet adaptors, specifies whether the device should be connected at power on.
Parent	Specifies the InstanceID of the parent controller (if any).
Connection	Used with Ethernet adaptors to specify the network connection name for the virtual system.
Address	Is device specific.
AddressOnParent	For a device, specifies its location on the controller.
AllocationUnits	Specifies the unit of allocation used.
VirtualQuantity	Specifies the quantity of a resource presented.
Reservation	Specifies the minimum quantity of a resource guaranteed to be available.
Limit	Specifies the maximum quantity of a resource that is granted.
Weight	Specifies a relative priority for this allocation in relation to other allocations.

717 Only fields directly related to describing devices are mentioned. Refer to the CIM MOF for a complete
718 description of all fields, each field corresponds to the identically named property in the
719 `CIM_ResourceAllocationSettingData` class or a class derived from it.

720 8.4 Ranges on elements

721 The optional `ovf:bound` attribute may be used to specify ranges for the `Item` elements. A range has a
722 minimum, normal, and maximum value, denoted by `min`, `normal`, and `max`, where `min <= normal <=`
723 `max`. The default values for `min` and `max` are those specified for `normal`.

724 See ANNEX F for virtual hardware deployment considerations.

725 For the `Item`, `EthernetPortItem`, and `StorageItem` elements in the `VirtualHardwareSection` and
726 the `ResourceAllocationSection` elements, the following additional semantics are defined:

- 727 • Each `Item`, `EthernetPortItem`, or `StorageItem` element has an optional `ovf:bound`
728 attribute. This value may be specified as `min`, `max`, or `normal`. The value defaults to `normal`.

- 729 • If the `ovf:bound` value is specified as either `min` or `max`, the item is used to specify the upper or
730 lower bound for one or more values for a given `InstanceID`. Such an item is called a range
731 marker.

732 The semantics of range markers are as follows:

- 733 • `InstanceID` and `ResourceType` shall be specified, and the `ResourceType` shall match other
734 Item elements with the same `InstanceID`.
- 735 • No more than one `min` range marker and no more than one `max` range marker for a given
736 RASD, EPASD, or SASD (identified with `InstanceID`) shall be specified.
- 737 • An `Item`, `EthernetPortItem`, or `StorageItem` element with a range marker shall have a
738 corresponding `Item`, `EthernetPortItem`, or `StorageItem` element without a range marker;
739 that is, an `Item`, `EthernetPortItem`, and `StorageItem` element with no `ovf:bound` attribute
740 or `ovf:bound` attribute with value `normal`. This corresponding item specifies the default value.
- 741 • For an `Item`, `EthernetPortItem`, and `StorageItem` element where only a `min` range marker
742 is specified, the `max` value is unbounded upwards within the set of valid values for the property.
- 743 • For an `Item`, `EthernetPortItem`, and `StorageItem` where only a `max` range marker is
744 specified, the `min` value is unbounded downwards within the set of valid values for the property.
- 745 • The default value shall be inside the range.
- 746 • Non-integer elements shall not be used in the range markers for RASD, EPASD, or SASD.

747 See D.9 for an example.

748

749 **9 Core metadata sections**750 Table 5 shows the core metadata sections that are defined in the `ovf` namespace.751 **Table 5 – Core metadata sections**

Section element	Parent element	Multiplicity
<code>DiskSection</code> Describes meta-information about all virtual disks in the package	Envelope	Zero or one
<code>NetworkSection</code> Describes logical networks used in the package	Envelope	Zero or one
<code>ResourceAllocationSection</code> Specifies reservations, limits, and shares on a given resource, such as memory or CPU for a virtual system collection	VirtualSystemCollection	Zero or one
<code>AnnotationSection</code> Specifies a free-form annotation on an entity	VirtualSystem VirtualSystemCollection	Zero or one
<code>ProductSection</code> Specifies product-information for a package, such as product name and version, along with a set of properties that can be configured	VirtualSystem VirtualSystemCollection	Zero or more
<code>EulaSection</code> Specifies a license agreement for the software in the package	VirtualSystem VirtualSystemCollection	Zero or more
<code>StartupSection</code> Specifies how a virtual system collection is powered on	VirtualSystemCollection	Zero or one
<code>DeploymentOptionSection</code> Specifies a discrete set of intended resource requirements	Envelope	Zero or one
<code>OperatingSystemSection</code> Specifies the guest software installed in a virtual system	VirtualSystem	Zero or one
<code>InstallSection</code> Specifies that the virtual system needs to be initially booted to install and configure the software	VirtualSystem	Zero or one
<code>EnvironmentFilesSection</code> Specifies additional files from an OVF package to be included in the OVF environment	VirtualSystem	Zero or one
<code>BootDeviceSection</code> Specifies boot device order to be used by a virtual system	VirtualSystem	Zero or more
<code>SharedDiskSection</code> Specifies virtual disks shared by more than one VirtualSystems at runtime	Envelope	Zero or one
<code>ScaleOutSection</code> Specifies that a VirtualSystemCollection contain a set of children that are homogeneous with respect to a prototype	VirtualSystemCollection	Zero or more
<code>PlacementGroupSection</code> Specifies a placement policy for a group of VirtualSystems or VirtualSystemCollections	Envelope	Zero or more
<code>PlacementSection</code> Specifies membership of a particular placement policy group	VirtualSystem VirtualSystemCollection	Zero or one

Section element	Parent element	Multiplicity
EncryptionSection Specifies encryption scheme for encrypting parts of an OVF descriptor or files to which it refers.	Envelope	Zero or one

752 The following subclauses describe the semantics of the core sections and provide some examples. The
753 sections are used in several places of an OVF envelope; the description of each section defines where it
754 may be used. See the [DSP8023](#) schema for a detailed specification of all attributes and elements.

755 In the OVF Schema, all sections are part of a substitution group with the `Section` element as head of the
756 substitution group. The `Section` element is abstract and cannot be used directly.

757 9.1 DiskSection

758 The `DiskSection` element describes meta-information about the virtual disks in the OVF package. The
759 virtual disks and associated metadata are described outside of the `VirtualHardwareSection` element to
760 facilitate sharing between the virtual systems within an OVF package.

761 The virtual disks in the `DiskSection` element may be referenced by one or more virtual systems.
762 However, as seen from the guest software, each virtual system gets individual private disks. Any level of
763 sharing done at runtime is virtualization platform specific and not visible to the guest software. See clause
764 9.13 for details about how to configure sharing of a virtual disk at runtime with concurrent access. See
765 D.10 for an example.

766 The `DiskSection` element is only valid as a direct child element of the `Envelope` element.

767 Each virtual disk represented by a `Disk` element shall be given an identifier using the `ovf:diskId`
768 attribute; the identifier shall be unique within the `DiskSection` element.

769 The capacity of a virtual disk shall be specified by the `ovf:capacity` attribute with an `xs:long` integer
770 value. The default unit of allocation shall be bytes. The optional string attribute
771 `ovf:capacityAllocationUnits` may be used to specify a particular unit of allocation. Values for
772 `ovf:capacityAllocationUnits` shall match the format for programmatic units defined in [DSP0004](#) with
773 the restriction that the base unit shall be "byte".

774 The `ovf:fileRef` attribute denotes the virtual disk content by identifying an existing `File` element in the
775 `References` element. The `File` element is identified by matching its `ovf:id` attribute value with the
776 `ovf:fileRef` attribute value. Omitting the `ovf:fileRef` attribute shall indicate an empty disk. If an empty
777 disk is indicated, the virtual disk shall be created and the content zeroed at deployment.

778 The format URI (see 5.2) of a non-empty virtual disk shall be specified by the `ovf:format` attribute.

779 Different `Disk` elements shall not contain `ovf:fileRef` attributes with identical values. `Disk` elements
780 shall be ordered such that they identify any `File` elements in the same order as these are defined in the
781 `References` element.

782 For empty disks, rather than specifying a fixed virtual disk capacity, the capacity may be given using a
783 reference to a `Property` element in a `ProductSection` element. This is done by setting
784 `ovf:capacity="{<id>}"` where `<id>` shall be the identifier of a `Property` element in the
785 `ProductSection` element. The `Property` element value shall resolve to an `xs:long` integer value. See
786 9.5 for a description of `Property` elements. The `ovf:capacityAllocationUnits` attribute is useful
787 when using `Property` elements because a user may be prompted and can then enter disk sizing
788 information in appropriate units, for example gigabytes.

789 For non-empty disks, the actual used size of the disk may be specified using the `ovf:populatedSize`
790 attribute. The unit of this attribute shall be bytes. The `ovf:populatedSize` attribute may be an estimate
791 of used disk size but shall not be larger than `ovf:capacity`.

792 In `VirtualHardwareSection`, virtual disk devices may have a `rasd:HostResource` element referring to a
793 `Disk` element in `DiskSection`; see 8.3. The virtual disk capacity shall be defined by the `ovf:capacity`
794 attribute on the `Disk` element. If a `rasd:VirtualQuantity` element is specified along with the
795 `rasd:HostResource` element, the virtual quantity value shall not be considered and may have any value.

796 A disk image may be represented as a set of modified blocks in comparison to a parent image. The use
797 of parent disks can often significantly reduce the size of an OVF package if it contains multiple disks with
798 similar content, such as a common base operating system. See ANNEX F for deployment considerations.

799 For the `Disk` element, a parent disk may be specified using the `ovf:parentRef` attribute that shall
800 contain a valid `ovf:diskId` reference to a different `Disk` element. If a disk block does not exist locally,
801 lookup for that disk block then occurs in the parent disk. In `DiskSection`, parent `Disk` elements shall
802 occur before child `Disk` elements that refer to them. Similarly, in `References` element, the `File` elements
803 referred from these `Disk` elements shall respect the same ordering. The ordering restriction ensures that
804 parent disks always occur before child disks, making it possible for a tool to consume the OVF package in
805 a streaming mode; see also clause 5.3.

806 9.2 NetworkSection

807 The `NetworkSection` element shall list all logical networks used in the OVF package. See D.11 for an
808 example.

809 The `NetworkSection` is only valid as a direct child element of the `Envelope` element. A `Network` element
810 is a child element of `NetworkSection`. Each `Network` element in the `NetworkSection` shall be given a
811 unique name using the `ovf:name` attribute. The name shall be unique within an OVF envelope.

812 All networks referred to from `Connection` elements in all `VirtualHardwareSection` elements shall be
813 defined in the `NetworkSection`.

814 Each logical network may contain a set of networking attributes that should be applied when mapping the
815 logical network at deployment time to a physical or virtual network. Networking attributes are specified by
816 zero or more instances of `NetworkPortProfile` child element or `NetworkPortProfileURI` child
817 element of the `Network` element.

818 The `NetworkPortProfile` element shall contain zero or more instances of `Item` elements of type
819 `epasd:CIM_EthernetPortAllocationSettingData_Type` that define the contents of zero or more
820 network port profiles. The `NetworkPortProfileURI` shall be a URI reference to a network port profile.

821 Examples of using the network port profiles are in ANNEX E.

822 9.3 ResourceAllocationSection

823 The `ResourceAllocationSection` element describes all resource allocation requirements of a
824 `VirtualSystemCollection` entity and applies only to the direct child `VirtualSystem` elements that do
825 not contain a `VirtualHardwareSection` element. It does not apply to a child `VirtualSystemCollection`
826 elements.

827 See ANNEX F for deployment considerations. See D.12 for an example.

828 The `ResourceAllocationSection` is a valid element for a `VirtualSystemCollection` entity.

829 The `ovf:configuration` attribute is optional and contains a list of configuration names. See 9.8 on
830 deployment options for semantics of this attribute.

831 The `ovf:bound` attribute is optional and contains a value of `min`, `max`, or `normal`. See 8.4 for semantics of
832 this attribute.

833 9.4 AnnotationSection

834 The `AnnotationSection` element is a user-defined annotation on an entity. See ANNEX F for
835 deployment considerations. See D.13 for an example.

836 The `AnnotationSection` element is a valid element for the `VirtualSystem` and the
837 `VirtualSystemCollection` entities.

838 See clause 10 for details about how to localize the `Annotation` element.

839 9.5 ProductSection

840 The `ProductSection` element specifies product-information for an appliance, such as product name,
841 version, and vendor. Typically it corresponds to a particular software product that is installed.

842 Zero or more elements may be specified within a `VirtualSystem` element or
843 `VirtualSystemCollection` element.

844 Each `ProductSection` element with the same parent element shall have a unique `ovf:class` and
845 `ovf:instance` attribute pair. If there is only one `ProductSection` element, the `ovf:class` and
846 `ovf:instance` attributes are optional and default to an empty string.

847 The `ovf:class` attribute should be used to identify the software product using the reverse domain name
848 convention. Examples of values are `com.vmware.tools` and `org.apache.tomcat`. If multiple instances of the
849 same product are installed, the `ovf:instance` attribute shall be used to identify the different instances.

850 If a `ProductSection` element exists, the first `ProductSection` element defined in the `VirtualSystem`
851 element or `VirtualSystemCollection` element that is the direct child element of the root element of an
852 OVF package shall define summary information that describes the entire package. This information may
853 be mapped into an instance of the `CIM_Product` class.

854 See D.14 for an example.

855 The `Product` element is optional and specifies the name of the product.

856 The `Vendor` element is optional and specifies the name of the product vendor.

857 The `Version` element is optional and specifies the product version in short form.

858 The `FullVersion` element is optional and describes the product version in long form.

859 The `ProductUrl` element is optional and specifies a URL that shall resolve to a human-readable
860 description of the product.

861 The `VendorUrl` element is optional and specifies a URL that shall resolve to a human-readable
862 description of the vendor.

863 The `AppUrl` element is optional and specifies a URL resolving to the deployed product instance.

864 The `Icon` element is optional and specifies display icons for the product.

865 9.5.1 Property elements

866 The `Property` elements specify customization parameters and are relevant to appliances that need to be
867 customized during deployment with specific settings such as network identity, the IP addresses of DNS
868 servers, gateways, and others.

869 The `ProductSection` is a valid section for a `VirtualSystem` and a `VirtualSystemCollection` entity.

870 The `Property` elements may be grouped by using `Category` elements. The set of `Property` elements
871 grouped by a `Category` element is the sequence of `Property` elements following the `Category` element,
872 until but not including an element that is not a `Property` element. For OVF packages containing a large
873 number of `Property` elements, this may provide a simpler installation experience. Similarly, each
874 `Property` element may have a short label defined by its `Label` child element in addition to a description
875 defined by its `Description` child element. See clause 10 for details about how to localize the `Category`
876 element and the `Description` and `Label` child elements of the `Property` element.

877 Each `Property` element in a `ProductSection` shall be given an identifier that is unique within the
878 `ProductSection` using the `ovf:key` attribute. The `ovf:key` attribute shall not contain the period character
879 ('.') or the colon character (':')

880 Each `Property` element in a `ProductSection` shall be given a type using the `ovf:type` attribute and
881 optionally type qualifiers using the `ovf:qualifiers` attribute. Valid types are listed in Table 6, and valid
882 qualifiers are listed in Table 7.

883 The optional attribute `ovf:value` is used to provide a default value for a `Property` element. One or more
884 optional `Value` elements may be used to define alternative default values for different configurations, as
885 defined in 9.8.

886 The optional attribute `ovf:userConfigurable` determines whether the property value is configurable
887 during the installation phase. If `ovf:userConfigurable` is `FALSE` or omitted, the `ovf:value` attribute
888 specifies the value to be used for that customization parameter during installation. If
889 `ovf:userConfigurable` is `TRUE`, the `ovf:value` attribute specifies a default value for that customization
890 parameter, which may be changed during installation.

891 A simple OVF implementation, such as a command-line installer, typically uses default values for
892 properties and does not prompt even though `ovf:userConfigurable` is set to `TRUE`. To force prompting
893 at startup time, omitting the `ovf:value` attribute is sufficient for integer types, because the empty string is
894 not a valid integer value. For string types, prompting may be forced by adding a qualifier requiring a non-
895 empty string; see Table 7.

896 The `ovf:password` attribute indicates that the property value may contain sensitive information. The
897 default value is `FALSE`. OVF implementations prompting for property values are advised to obscure these
898 values when the `ovf:password` attribute is set to `TRUE`. Note that this mechanism affords limited security
899 protection only. Although sensitive values are masked from casual observers, default values in the OVF
900 descriptor and assigned values in the OVF environment are still passed in clear text.

901 The ID and the value of the `Property` elements are exposed to the guest software using the OVF
902 environment file. The `ovf:class` and `ovf:instance` attributes shall not contain the colon character (':'). If only
903 one instance of a product is installed, the `ovf:instance` attribute should not be set. The value of the
904 `ovfenv:key` attribute of a `Property` element exposed in the OVF environment shall be constructed from
905 the value of the `ovf:key` attribute of the corresponding `Property` element defined in a `ProductSection`
906 entity of an OVF descriptor as follows:

```
907 key-value-env = [class-value "."] key-value-prod ["." instance-value]
```

908 The syntax definition above use ABNF with the exceptions listed in ANNEX A, where:

- 909 • `class-value` is the value of the `ovf:class` attribute of the `Property` element defined in the
910 `ProductSection` entity. The production [`class-value` "."] shall be present if and only if `class-`
911 `value` is not the empty string.
- 912 • `key-value-prod` is the value of the `ovf:key` attribute of the `Property` element defined in the
913 `ProductSection` entity.
- 914 • `instance-value` is the value of the `ovf:instance` attribute of the `Property` element defined in the
915 `ProductSection` entity. The production ["." `instance-value`] shall be present if and only if
916 `instance-value` is not the empty string.

917 If the `ovf:userConfigurable` attribute is TRUE, the deployment function should prompt for values of the
918 `Property` elements. These `Property` elements may be defined in multiple `ProductSection` elements.

919 `Property` elements specified on a `VirtualSystemCollection` element are also seen by its immediate
920 child elements. Child elements may refer to the properties of a parent `VirtualSystemCollection`
921 element using macros on the form `$(name)` as value for `ovf:value` attributes.

922 Table 6 lists the valid types for properties. These are a subset of CIM intrinsic types defined in [DSP0004](#)
923 that also define the value space and format for each intrinsic type. Each `Property` element shall specify a
924 type using the `ovf:type` attribute.

925 **Table 6 – Property types**

Type	Description
uint8	Unsigned 8-bit integer
sint8	Signed 8-bit integer
uint16	Unsigned 16-bit integer
sint16	Signed 16-bit integer
uint32	Unsigned 32-bit integer
sint32	Signed 32-bit integer
uint64	Unsigned 64-bit integer
sint64	Signed 64-bit integer
String	String
Boolean	Boolean
real32	IEEE 4-byte floating point
real64	IEEE 8-byte floating point

926 Table 7 lists the supported CIM type qualifiers as defined in [DSP0004](#). Each `Property` element may
927 optionally specify type qualifiers using the `ovf:qualifiers` attribute with multiple qualifiers separated by
928 commas; see production `qualifierList` in ANNEX A “MOF Syntax Grammar Description” in [DSP0004](#).

929

Table 7 – Property qualifiers

Property Type	Property Qualifier
String	MinLen (min) MaxLen (max) ValueMap{...}
uint8 sint8 uint16 sint16 uint32 sint32 uint64 sint64	ValueMap{...}

930 9.6 EulaSection

931 A `EulaSection` contains the legal terms for using its parent `Content` element. Multiple `EulaSections`
 932 may be present in an OVF. See ANNEX F for deployment considerations. See D.15 for an example. The
 933 `EulaSection` is a valid section for a `VirtualSystem` and a `VirtualSystemCollection` entity.

934 See clause 10 for details about how to localize the `License` element.

935 See also clause 10 for a description of storing EULA license contents in an external file without any XML
 936 header or footer. This allows inclusion of standard license or copyright text files in unaltered form.

937 9.7 StartupSection

938 The `StartupSection` element specifies how a collection of virtual systems identified by a
 939 `VirtualSystemCollection` element is powered on and off. The `StartupSection` element shall not be
 940 part of a `VirtualSystem` element. See D.16 for an example.

941 If a `VirtualSystemCollection` element has a `StartupSection` element then each `VirtualSystem`
 942 element or `VirtualSystemCollection` element that is a direct child element shall have a corresponding
 943 `Item` element in the `StartupSection` element.

944 When a start or stop action is performed on a `VirtualSystemCollection` element, the respective actions
 945 on the `Item` elements of its `StartupSection` element are invoked in the specified order. Whenever an
 946 `Item` element corresponds to a nested `VirtualSystemCollection` element, the actions on the `Item`
 947 elements of its `StartupSection` element shall be invoked before the action on the `Item` element
 948 corresponding to that `VirtualSystemCollection` element is invoked (i.e., depth-first traversal).

949 The following required attributes on `Item` element are supported for a `VirtualSystem` and
 950 `VirtualSystemCollection` elements:

- 951 • `ovf:id` shall match the value of the `ovf:id` attribute of a `Content` element which is a direct
 952 child of this `VirtualSystemCollection`. That `Content` element describes the virtual system or
 953 virtual system collection to which the actions defined in the `Item` element apply.
- 954 • `ovf:order` specifies the startup order of the item using non-negative integer values. If the
 955 `ovf:order = "0"`, the order is not specified. If the `ovf:order` is non-zero, the of execution of
 956 the start action shall be the numerical ascending order of the values. The `Items` with same
 957 order identifier may be started concurrently.

958 The order of execution of the stop action should be the numerical descending order of the values if the
 959 `ovf:shutdownorder` attribute is not specified. In implementation-specific scenarios, the order of
 960 execution of the stop action may be non-descending.

961 The following optional attributes on the `Item` element are supported for a `VirtualSystem` element.

- 962 • `ovf:shutdownorder` specifies the shutdown order using non-negative integer values. If the
 963 `ovf:shutdownorder = "0"`, the shutdown order is not specified. If the `ovf:shutdownorder` is
 964 non-zero, the order of execution of the stop action shall be the numerical descending order of
 965 the values. The `Items` with same order identifier may be stopped concurrently.
- 966 • `ovf:startDelay` specifies a delay in seconds to wait until proceeding to the next virtual system
 967 in the start sequence. The default value is 0.
- 968 • `ovf:waitingForGuest` enables the virtualization platform to resume the startup sequence after
 969 the guest software has reported it is ready. The interpretation of this is virtualization platform
 970 specific. The default value is FALSE.
- 971 • `ovf:startAction` specifies the start action to use. Valid values are `powerOn` and `none`. The
 972 default value is `powerOn`.
- 973 • `ovf:stopDelay` specifies a delay in seconds to wait until proceeding to the previous order in the
 974 stop sequence. The default value is 0.
- 975 • `ovf:stopAction` specifies the stop action to use. Valid values are `powerOff`, `guestShutdown`,
 976 and `none`. The interpretation of `guestShutdown` is virtualization platform specific. The default
 977 value is `powerOff`.

978 If the `StartupSection` element is not specified, an `ovf:order="0"` attribute is implied.

979 9.8 DeploymentOptionSection

980 The `DeploymentOptionSection` element specifies a discrete set of intended resource configurations.
 981 The author of an OVF package can include sizing metadata for different configurations. The deployment
 982 shall select one of the configurations, e.g., by prompting the user. The selected configuration shall be
 983 available in the OVF environment file. See ANNEX F.

984 The `DeploymentOptionSection` specifies an ID, label, and description for each configuration. See D.17
 985 for an example.

986 The `DeploymentOptionSection` has the following semantics:

- 987 • If present, the `DeploymentOptionSection` is valid only as a direct child element of the root
 988 element. Only one `DeploymentOptionSection` section shall be present in an OVF descriptor.
- 989 • The discrete set of configurations is described with `Configuration` elements, which shall have
 990 identifiers specified by the `ovf:id` attribute that are unique in the OVF package.
- 991 • A default `Configuration` element may be specified with the optional `ovf:default` attribute.
 992 Only one default `Configuration` element shall be specified. If no default is specified, the first
 993 element in the list is the default.
- 994 • The `Label` and `Description` elements are localizable using the `ovf:msgid` attribute. See
 995 clause 10 for more details about internationalization support.

996 Configurations may be used to control resources for virtual hardware and for virtual system collections.
 997 The `Item`, `EthernetPortItem`, and `StorageItem` elements in `VirtualHardwareSection` elements
 998 describe resources for `VirtualSystem` entities, while the `Item`, `EthernetPortItem`, and `StorageItem`
 999 elements in `ResourceAllocationSection` elements describe resources for virtual system collections. For

1000 these two `Item`, `EthernetPortItem`, or `StorageItem` types, the following additional semantics are
 1001 defined:

- 1002 • Each `Item`, `EthernetPortItem`, and `StorageItem` has an optional `ovf:configuration`
 1003 attribute, containing a list of configurations separated by a single space character. If not
 1004 specified, the item shall be selected for any configuration. If specified, the item shall be selected
 1005 only if the chosen configuration ID is in the list. A configuration attribute shall not contain a
 1006 configuration ID that is not specified in the `DeploymentOptionSection`.
- 1007 • Within a single `VirtualHardwareSection` or `ResourceAllocationSection`, multiple `Item`,
 1008 `EthernetPortItem`, and `StorageItem` elements are allowed to refer to the same `InstanceID`. A
 1009 single combined `Item`, `EthernetPortItem`, or `StorageItem` for the given `InstanceID` shall be
 1010 constructed by picking up the child elements of each `Item`, `EthernetPortItem`, or `StorageItem`
 1011 element, with child elements of a former `Item`, `EthernetPortItem`, or `StorageItem` element in
 1012 the OVF descriptor not being picked up if there is a like-named child element in a latter `Item`,
 1013 `EthernetPortItem`, or `StorageItem` element. Any attributes specified on child elements of
 1014 `Item`, `EthernetPortItem`, or `StorageItem` elements that are not picked up that way, are not
 1015 part of the combined `Item`, `EthernetPortItem`, or `StorageItem` element.
- 1016 • All `Item`, `EthernetPortItem`, `StorageItem` elements shall specify `ResourceType`, and `Item`,
 1017 `EthernetPortItem`, and `StorageItem` elements with the same `InstanceID` shall agree on
 1018 `ResourceType`.

1019 Note that the attributes `ovf:configuration` and `ovf:bound` on `Item` may be used in combination to
 1020 provide flexible configuration options.

1021 Configurations can further be used to control default values for properties and whether properties are
 1022 user configurable. For `Property` elements inside a `ProductSection`, the following additional semantic is
 1023 defined:

- 1024 • It is possible to specify alternative default property values for different configurations in a
 1025 `DeploymentOptionSection`. In addition to a `Label` and `Description` element, each `Property`
 1026 element may optionally contain `Value` elements. The `Value` element shall have an `ovf:value`
 1027 attribute specifying the alternative default and an `ovf:configuration` attribute specifying the
 1028 configuration in which this new default value should be used. Multiple `Value` elements shall not
 1029 refer to the same configuration.
- 1030 • A `Property` element may optionally have an `ovf:configuration` attribute specifying the
 1031 configuration in which this property should be user configurable. The value of
 1032 `ovf:userConfigurable` is implicitly set to `FALSE` for all other configurations, in which case the
 1033 default value of the property may not be changed during installation.

1034 9.9 OperatingSystemSection

1035 An `OperatingSystemSection` specifies the operating system installed on a virtual system. See D.18 for
 1036 an example.

1037 The values for `ovf:id` should be taken from the `ValueMap` of the `CIM_OperatingSystem.OsType`
 1038 property. The description should be taken from the corresponding `Values` of the
 1039 `CIM_OperatingSystem.OsType` property.

1040 The `OperatingSystemSection` element is a valid section for a `VirtualSystem` element only.

1041 9.10 InstallSection

1042 The `InstallSection` element, if specified, indicates that the virtual system needs to be booted once in
 1043 order to install and/or configure the guest software. The guest software is expected to access the OVF

- 1044 environment during that boot, and to shut down after having completed the installation and/or
1045 configuration of the software, powering off the guest.
- 1046 If the `InstallSection` is not specified, this indicates that the virtual system does not need to be powered
1047 on to complete installation of guest software. See D.19 for an example.
- 1048 The `InstallSection` element shall be valid only for a `VirtualSystem` element.
- 1049 The `ovf:initialBootStopDelay` attribute specifies a delay in seconds to wait for the virtual system to
1050 power off.
- 1051 If the delay expires and the virtual system has not powered off, the deployment function shall indicate a
1052 failure.
- 1053 An `ovf:initialBootStopDelay` attribute value of zero indicates that the boot stop delay is not specified.
- 1054 Note that the guest software in the virtual system can do multiple reboots before powering off.
- 1055 Several virtual systems in a virtual system collection may have an `InstallSection` element defined, in
1056 which case the above step is done for each virtual system that has an `InstallSection` element.

1057 9.11 EnvironmentFilesSection

- 1058 The `EnvironmentFilesSection` enables the OVF package to specify additional environment file(s) (AEF)
1059 besides the virtual disks. These AEFs enable increased flexibility in image customization outside of virtual
1060 disk capture, allowing an OVF package to provide customized solutions by combining existing virtual
1061 disks without modifying them.
- 1062 The AEF contents are neither generated nor validated by the deployment function.
- 1063 The AEFs are included in the transport media generated by the deployment function.
- 1064 The AEFs are conveyed to the guest software using the indicated transport media type. The AEFs and
1065 OVF environment files are intended to use same transport media and transport media type
- 1066 The `EnvironmentFilesSection` shall contain a `File` element with the attributes `ovf:fileRef` and
1067 `ovf:path` for each AEF provided to the guest software.
- 1068 The `ovf:fileRef` attribute shall specify an existing `File` element in the `References` element. The `File`
1069 element is identified by matching its `ovf:id` attribute value with the `ovf:fileRef` attribute value.
- 1070 The `ovf:path` attribute specifies the relative location in the transport media (see clause 11.1) where the
1071 file should be placed, using the syntax of relative-path references in [RFC3986](#).
- 1072 The referenced `File` element in the `References` element identifies the content using one of the URL
1073 schemes "file", "http", or "https". For the "file" scheme, the content is static and included in the
1074 OVF package. See ANNEX F for deployment considerations
- 1075 For details about transport media type, see clause 11.2.

1076 9.12 BootDeviceSection

- 1077 Individual virtual systems use the default device boot order provided by the virtualization platform's virtual
1078 BIOS. The `BootDeviceSection` allows the OVF package author to specify particular boot configurations
1079 and boot order settings. This enables booting from non-default devices, such as a NIC using PXE, a USB
1080 device, or a secondary disk. Moreover, there could be multiple boot configurations with different boot
1081 orders. For example, a virtual disk may need to be patched before it is bootable and a patch ISO image
1082 could be included in the OVF package.

1083 The Common Information Model (CIM) defines artifacts to deal with boot order use cases prevalent in the
1084 industry for BIOSes found in desktops and servers. The boot configuration is defined by the class
1085 `CIM_BootConfigSetting` that in turn contains one or more `CIM_BootSourceSetting` classes as defined
1086 in the CIM Schema. Each class representing the boot source in turn has either the specific device or a
1087 “device type”, such as disk or CD/DVD, as a boot source.

1088 In the context of this specification, the `InstanceID` property of `CIM_BootSourceSetting` is used for
1089 identifying a specific device as the boot source. The `InstanceID` property of the device as specified in the
1090 `Item` description of the device in the `VirtualHardwareSection` element is used to specify the device as
1091 a boot source. In case the source is desired to be a device type, the `StructuredBootString` field is
1092 used to denote the type of device with values defined by the CIM boot control profile. See ANNEX F for
1093 deployment considerations.

1094 See D.21 for an example.

1095 **9.13 SharedDiskSection**

1096 The existing `DiskSection` element in clause 9.1 describes virtual disks in the OVF package. Virtual disks
1097 in the `DiskSection` element can be referenced by multiple virtual systems, but seen from the guest
1098 software, each virtual system gets individual private disks. Any level of sharing done at runtime is
1099 virtualization platform specific and not visible to the guest software.

1100 Certain applications, such as clustered databases, rely on multiple virtual systems sharing the same
1101 virtual disk at runtime. `SharedDiskSection` allows the OVF package to specify `Disk` elements shared by
1102 more than one virtual system at runtime. These virtual disks may be backed by an external `File`
1103 reference, or may be empty virtual disks without backing. It is recommended that the guest software use
1104 cluster-aware file system technology to be able to handle concurrent access. See D.22 for an example.

1105 The `SharedDiskSection` is a valid section at the outermost envelope level only.

1106 Each virtual disk is represented by a `SharedDisk` element that shall be given an identifier using the
1107 `ovf:diskId` attribute; the identifier shall be unique within the combined content of `DiskSection` and
1108 `SharedDiskSection` element. The `SharedDisk` element has the same structure as the `Disk` element in
1109 the `DiskSection` element, with the addition of an `ovf:readOnly` attribute. The `ovf:readOnly` is optional
1110 and states whether shared disk access is read-write, i.e., `FALSE`, or read-only, i.e., `TRUE`.

1111 Shared virtual disks are referenced from virtual hardware by using the `HostResource` element as
1112 described in clause 8.3.

1113 Support of the `SharedDiskSection` element is optional. The virtualization platform should give an
1114 appropriate error message based on the value of the `ovf:required` attribute on the `SharedDiskSection`
1115 element.

1116 **9.14 ScaleOutSection**

1117 The number of virtual systems or collections of virtual system contained in an OVF package is fixed and
1118 determined by the structure inside the `Envelope` element. The `ScaleOutSection` element allows a
1119 `VirtualSystemCollection` element to contain a set of children that are homogeneous with respect to a
1120 prototypical `VirtualSystem` or `VirtualSystemCollection` element. The `ScaleOutSection` element
1121 shall cause the deployment function to replicate the prototype a number of times, thus allowing the
1122 number of instantiated virtual systems to be configured dynamically at deployment time. See D.23 for an
1123 example.

1124 This mechanism enables scaling of virtual system instances at deployment time. Scaling at runtime is not
1125 within the scope of this specification.

- 1126 The `ScaleOutSection` element is a valid section inside `VirtualSystemCollection` element only.
- 1127 The `ovf:id` attribute on `ScaleOutSection` element identifies the virtual system or collection of virtual
1128 systems prototype to be replicated.
- 1129 For the `InstanceCount` element, the `ovf:minimum` and `ovf:maximum` attribute values shall be non-
1130 negative integers and `ovf:minimum` shall be less than or equal to the value of `ovf:maximum`. The
1131 `ovf:minimum` value may be zero in which case the `VirtualSystemCollection` may contain zero
1132 instances of the prototype. If the `ovf:minimum` attribute is not present, it shall be assumed to have a value
1133 of one. If the `ovf:maximum` attribute is not present, it shall be assumed to have a value of unbounded.
1134 The `ovf:default` attribute is required and shall contain a value within the range defined by `ovf:minimum`
1135 and `ovf:maximum`.
- 1136 Each replicated instance shall be assigned a unique `ovf:id` value within the `VirtualSystemCollection`
1137 element. The unique `ovf:id` value shall be constructed from the prototype `ovf:id` value with a sequence
1138 number appended as follows:
- ```
1139 replica-ovf-id = prototype-ovf-id "-" decimal-number
1140 decimal-number = decimal-digit | (decimal-digit decimal-number)
1141 decimal-digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
```
- 1142 The syntax definitions above use ABNF with the exceptions listed in ANNEX A. The first replica shall  
1143 have sequence number one and following sequence numbers shall be incremented by one for each  
1144 replica. Note that after deployment, no `VirtualSystem` will have the prototype `ovf:id` value itself.
- 1145 If the prototype being replicated has a starting order in the `StartupSection`, all replicas shall share this  
1146 value. It is not possible to specify a particular starting sequence among replicas.
- 1147 Property values for `Property` elements in the prototype are prompted once per replica created. If the  
1148 OVF package author requires a property value to be shared among instances, that `Property` may be  
1149 declared at the containing `VirtualSystemCollection` level and referenced by replicas as described in  
1150 clause 9.5.
- 1151 Configurations from the `DeploymentOptionSection` element may be used to control values for  
1152 `InstanceCount` element. The `InstanceCount` element may have an `ovf:configuration` attribute  
1153 specifying the configuration in which this element should be used. Multiple elements shall not refer to the  
1154 same configuration, and a configuration attribute shall not contain an `ovf:id` value that is not specified in  
1155 the `DeploymentOptionSection`. See D.23 for an example.

## 1156 9.15 PlacementGroupSection and PlacementSection

- 1157 Guest software may require the deployment of virtual systems with specific proximity needs. There are  
1158 two use cases:
- 1159 1) the ability to specify that two or more virtual systems should be deployed closely together  
1160 because they rely on fast communication or have a common dependency
  - 1161 2) the ability to specify that two or more virtual systems should be deployed on different platforms  
1162 or locations because of high-availability or disaster recovery considerations
- 1163 The `PlacementGroupSection` element allows an OVF package to define a placement policy for a group  
1164 of `VirtualSystems`. The `PlacementSection` element allows the annotation of the elements with  
1165 membership of a particular placement policy group.
- 1166 Zero or more `PlacementGroupSections` may be defined at the `Envelope` level. The `PlacementSection`  
1167 element may be declared at the `VirtualSystem` or `VirtualSystemCollection` level.

1168 Declaring a `VirtualSystemCollection` a member of a placement policy group applies transitively to all  
 1169 child `VirtualSystem` and child `Virtual System Collections` elements provided that no placement  
 1170 policies are specified for the child `VirtualSystem` or `VirtualSystemCollection`.

1171 If a parent `VirtualSystemCollection` and child `VirtualSystem(s)` and/or  
 1172 `VirtualSystemCollection(s)` both have placement policies, the placement policies of the child  
 1173 `VirtualSystems` and/or child `VirtualSystemCollections` should be applied first. Then placement  
 1174 policy of the parent `VirtualSystemCollection` should be applied.

1175 In the event that there is a conflict in the placement policy, the availability policy should override the  
 1176 affinity policy

1177 The `ovf:id` attribute in `PlacementGroupSection` is used to identify a placement policy. The value of the  
 1178 `ovf:id` attribute shall be unique within the OVF package.

1179 Placement policy group membership is specified using the `ovf:group` attribute in the  
 1180 `PlacementSection`. The value of the `ovf:group` attribute shall match the value of an `ovf:id` attribute in  
 1181 a `PlacementGroupSection`. The value of the `ovf:group` attribute shall be a comma-separated text string  
 1182 of placement policy attributes.

1183 This standard defines the placement policies "affinity" and "availability", specified with the required  
 1184 `ovf:policy` attribute on `PlacementGroupSection`.

1185 The set of attributes used for availability and affinity are defined in Table 8 and Table 9.

1186

**Table 8 – Availability attributes**

| Attribute               | Description                                                                 |
|-------------------------|-----------------------------------------------------------------------------|
| availability            | The virtual systems should be placed on different virtualization platforms. |
| availability-geographic | The virtual systems should be placed in different geographical areas.       |
| availability-site       | The virtual systems should be placed on different operator sites.           |
| availability-rack       | The virtual systems should be placed on different physical racks.           |
| availability-chassis    | The virtual systems should be placed on different physical chassis.         |
| availability-host       | The virtual systems should be placed on different physical hosts.           |

1187

1188

**Table 9 – Affinity Attributes**

| Attribute           | Description                                                                                                                                                        |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| affinity            | The virtual systems should be placed on the same virtualization platform.                                                                                          |
| affinity-geographic | The virtual systems should be placed in the same geographical area.                                                                                                |
| affinity-site       | The virtual systems should be placed on the same operator site.                                                                                                    |
| affinity-rack       | The virtual systems should be placed on the same physical rack.                                                                                                    |
| affinity-chassis    | The virtual systems should be placed on the same physical chassis.                                                                                                 |
| affinity-host       | The virtual systems should be placed in close proximity, i.e., on the same physical host or on hosts that have low latency and high bandwidth network connectivity |

1189 The placement policies that can be declared within a `PlacementGroupSection` are combinations of the  
 1190 availability and affinity attributes defined in Table 8 and Table 9. The placement policy is a single string  
 1191 represented by concatenating the valid placement policy combinations using commas as separators.  
 1192 Allowed combinations of affinity and availability attributes is defined in Table 10.

1193

**Table 10 – Allowed combinations of scoped affinity and availability**

| Valid Combinations      | availability order | affinity | affinity-geographic | affinity-site | affinity-rack | affinity-chassis | affinity-host |
|-------------------------|--------------------|----------|---------------------|---------------|---------------|------------------|---------------|
| availability            |                    | No       | Yes                 | Yes           | Yes           | Yes              | No            |
| availability-geographic | 5                  | Yes      | No                  | No            | No            | No               | No            |
| availability-site       | 4                  | Yes      | Yes                 | No            | No            | No               | No            |
| availability-rack       | 3                  | Yes      | Yes                 | Yes           | No            | No               | No            |
| availability-chassis    | 2                  | Yes      | Yes                 | Yes           | Yes           | No               | No            |
| availability-host       | 1                  | No       | Yes                 | Yes           | Yes           | Yes              | No            |

1194 The availability of the parent shall be higher availability order than the availability of the child.

1195 If the placement policy is 'availability' without scoping, no availability order is specified.

1196 See D.24 for an example.

1197 **9.16 EncryptionSection**

1198 It is desirable for licensing and other reasons to have an encryption scheme enabling free exchange of  
 1199 OVF appliances while ensuring that only the intended parties can use them. The XML Encryption Syntax  
 1200 and Processing standard is utilized to encrypt either the files in the reference section or any parts of the  
 1201 XML markup of an OVF document.

1202 The various aspects of OVF encryption are as shown below:

1203 1) block encryption

1204 The OVF package shall utilize block encryption algorithms as specified in the XML  
1205 encryption 1.1 documents (ref) for this purpose.

1206 2) key derivation

1207 The OVF package may use the appropriate key for this purpose. If the key is derived using  
1208 a passphrase, the author shall use one of the key derivations specified in the XML  
1209 Encryption 1.1 standard.

1210 3) key transport.

1211 If the encryption key is embedded in the OVF package, the specified key transport  
1212 mechanisms shall be used.

1213 This standard defines a section called the `EncryptionSection` as a focal point for the encryption  
1214 functionality. This section provides a single location for placing the encryption-algorithm-related markup  
1215 and the corresponding reference list to point to the OVF content that has been encrypted.

1216 Note that depending on the parts of the OVF package that has been encrypted, an OVF descriptor may  
1217 not validate against the [DSP8023](#) until decrypted. See D.25 for an example.

## 1218 10 Internationalization

1219 The following elements support localizable messages using the optional `ovf:msgid` attribute:

- 1220 • Info element on Content
- 1221 • Name element on Content
- 1222 • Info element on Section
- 1223 • Annotation element on AnnotationSection
- 1224 • License element on EulaSection
- 1225 • Description element on NetworkSection
- 1226 • Description element on OperatingSystemSection
- 1227 • Description, Product, Vendor, Label, and Category elements on ProductSection
- 1228 • Description and Label elements on Property
- 1229 • Description and Label elements on DeploymentOptionSection
- 1230 • ElementName, Caption and Description subelements on the System element in  
1231 VirtualHardwareSection
- 1232 • ElementName, Caption and Description subelements on Item elements in  
1233 VirtualHardwareSection
- 1234 • ElementName, Caption and Description subelements on Item elements in  
1235 ResourceAllocationSection

1236 The `ovf:msgid` attribute contains an identifier that refers to a message that may have different values in  
1237 different locales. See D.26 for an example.

1238 The `xml:lang` attribute on the `Envelope` element shall specify the default locale for messages in the  
1239 descriptor. The attribute is optional with a default value of "en-US".

## 1240 10.1 Internal resource bundles

1241 Message resource bundles can be internal or external to the OVF descriptor. Internal resource bundles  
1242 are represented as `Strings` elements at the end of the `Envelope` element. See D.26 for an example.

## 1243 10.2 External resource bundles

1244 External resource bundles shall be listed first in the `References` section and referred to from `Strings`  
1245 elements. An external message bundle follows the same schema as the embedded one. Exactly one  
1246 `Strings` element shall be present in an external message bundle, and that `Strings` element shall not  
1247 have an `ovf:fileRef` attribute specified. See D.26 for an example.

1248 The embedded and external `Strings` elements may be interleaved, but they shall be placed at the end of  
1249 the `Envelope` element. If multiple occurrences of a `msg:id` attribute with a given locale occur, a latter  
1250 value overwrites a former.

## 1251 10.3 Message content in external file

1252 The content of all localizable messages may be stored in an external file using the optional `ovf:fileRef`  
1253 attribute on the `Msg` element. For the `License` element on `EulaSection` in particular, this allows inclusion  
1254 of a standard license text file in unaltered form without any XML header or footer.

1255 The `ovf:fileRef` attribute denotes the message content by identifying an existing `File` element in the  
1256 `References` element; the `File` element is identified by matching its `ovf:id` attribute value with the  
1257 `ovf:fileRef` attribute value. The content of an external file referenced using `ovf:fileRef` shall be  
1258 interpreted as plain text in UTF-8 Unicode.

1259 If the referenced file is not available, the embedded content of the `Msg` element shall be used.

1260 The optional `ovf:fileRef` attribute may appear on `Msg` elements in both internal and external `Strings`  
1261 resource bundles. See D.27 for an example.

## 1262 11 OVF environment and OVF environment file

1263 The OVF environment defines how the guest software and the virtualization platform interact. The OVF  
1264 environment enables the guest software to access information about the virtualization platform, such as  
1265 the user-specified values for the properties defined in the OVF descriptor.

1266 [DSP8027](#) is the XML Schema definition file that contains the elements and attributes defining the format  
1267 and semantics of an XML document that constitutes the OVF environment file (OEF). The OEF shall  
1268 validate against [DSP8027](#).

1269 The OEF is created on a per virtual system basis by the deployment function. The basis of the OEF is the  
1270 OVF descriptor, OVF operational metadata, OVF property values, policy metadata, and other  
1271 user-provided values.

1272 The OEF provides the guest software information about the environment that it is being executed in. The  
1273 way that the OEF is conveyed depends on the transport media type. See D.28 for an example.

1274 The value of the `ovfenv:id` attribute of the `Environment` element shall match the value of the `ovf:id`  
1275 attribute of the `VirtualSystem` entity describing this virtual system.

1276 The `PlatformSection` element contains optional information provided by the deployment function. The  
 1277 `Kind`, `Version`, and `Vendor` elements describe the virtualization platform. The `Locale` and `TimeZone`  
 1278 elements describe the current locale and time zone.

1279 The `PropertySection` element contains `Property` elements with key/value pairs corresponding to all  
 1280 properties specified in the OVF descriptor for the current virtual system, as well as properties specified for  
 1281 the immediate parent `VirtualSystemCollection`, if one exists. The environment presents properties as  
 1282 a single list to make it easy for applications to parse. Furthermore, the single list format supports the  
 1283 override semantics that enables a property on a `VirtualSystem` to override a property defined on a  
 1284 parent `VirtualSystemCollection`. The property that is overridden shall not be in the list. If a property in  
 1285 a virtual system and a property in the parent `VirtualSystemCollection` have identical `ovf:key`,  
 1286 `ovf:class`, and `ovf:instance` attribute values the value of the parent property is overridden by the value  
 1287 of the child property; see 9.5. The value of the parent property may be obtained by adding a child  
 1288 property with a different name referencing the parent property with a macro; see 9.5.

1289 An `Entity` element shall exist for each sibling `VirtualSystem` and `VirtualSystemCollection`, if any  
 1290 are present. The value of the `ovfenv:id` attribute of the `Entity` element shall match the value of the  
 1291 `ovf:id` attribute of the sibling entity. The `Entity` elements contain the property key/value pairs in the  
 1292 sibling's OVF environment documents, so the content of an `Entity` element for a particular sibling shall  
 1293 contain the exact `PropertySection` seen by that sibling. This information can be used, for example, to  
 1294 make configuration information, such as IP addresses, available to `VirtualSystems` that are a part of a  
 1295 multitiered application.

1296 Table 11 shows the core sections that are defined.

1297 **Table 11 – Core sections for OEF**

| Section                                                                                                            | Location              | Multiplicity |
|--------------------------------------------------------------------------------------------------------------------|-----------------------|--------------|
| <code>PlatformSection</code><br>Provides information from the deployment platform                                  | Environment           | Zero or one  |
| <code>PropertySection</code><br>Contains key/value pairs corresponding to properties defined in the OVF descriptor | Environment<br>Entity | Zero or one  |

1298 The OEF is extensible by providing new section types. The deployment function should ignore unknown  
 1299 section types and elements specified in OEF.

## 1300 11.1 Transport media

1301 The transport media refers to the format used to convey the information to the guest software. The  
 1302 transport media (e.g., ISO image) is generated by the deployment function.

1303 If the transport media type is 'iso', the generated ISO image shall comply with the [ISO 9660](#) specification  
 1304 with support for Joliet extensions.

1305 The transport media shall contain the OVF environment file and any additional environment file(s) for this  
 1306 particular virtual system. The OEF shall be presented as an XML file named `ovf-env.xml` that is  
 1307 contained in the root directory of the transport media. The guest software is now able to access the  
 1308 information.

1309 For additional environment files, the transport media shall have the root location relative to the `ovf:path`  
 1310 attribute in a directory named "ovfiles" contained in the root directory. This provides an access  
 1311 mechanism for the guest software.



1312 Other custom transport media may support this mechanism. Custom transport medium shall specify how  
1313 to access multiple data sources from a root location. See D.20 for an example. The access mechanism  
1314 for the guest software is not specified.

## 1315 11.2 Transport media type

1316 The transport media type refers to a mechanism to convey transport media over a data link or removable  
1317 storage medium (e.g., CD/DVD-ROM) from deployment functions to guest software.

1318 The `iso` transport media type shall support this mechanism.

1319 This standard defines the “iso” transport type to meet the need for interoperability.

1320 The transport media can be communicated in a number of ways to the guest software. These ways are  
1321 called transport media types. The transport media types are specified in the OVF descriptor by the  
1322 `ovf:transport` attribute of `VirtualHardwareSection`. Several transport media types may be specified,  
1323 separated by a single space character, in which case an implementation is free to use any of them.

1324 To enable interoperability, this specification defines an `iso` transport media type, which all  
1325 implementations that support CD-ROM devices are required to support. The `iso` transport media type  
1326 communicates the environment document by making a dynamically generated ISO image available to the  
1327 guest software.

1328 To support the `iso` transport media type, prior to booting a virtual system, an implementation shall make  
1329 an ISO read-only disk image available as backing for a disconnected CD-ROM. If the `iso` transport media  
1330 type is selected for a `VirtualHardwareSection`, at least one disconnected CD-ROM device shall be  
1331 present in this section.

1332 If the virtual system prior to booting had more than one disconnected CD-ROM, the guest software may  
1333 have to scan connected CD-ROM devices in order to locate the ISO image containing the `ovf-env.xml`  
1334 file.

1335 The transport media containing the OVF environment file shall be made available to the guest software  
1336 on every boot of the virtual system.

1337 Support for the `iso` transport media type is not a requirement for virtual hardware architectures or guest  
1338 software that do not have CD-ROM device support.

1339 To be conformant with this specification, any transport media type other than `iso` shall be given by a URI  
1340 that identifies an unencumbered specification on how to use the transport media type. The specification  
1341 need not be machine readable, but it shall be static and unique so that it may be used as a key by  
1342 software reading an OVF descriptor to uniquely determine the transport media type. The specification  
1343 shall be sufficient for a skilled person to properly interpret the transport media type mechanism for  
1344 implementing the protocols. The URIs should be resolvable.

1345

## ANNEX A (informative)

### Symbols and conventions

1346  
1347  
1348  
1349

1350 XML examples use the XML namespace prefixes that are defined in Table 1. The XML examples use a  
1351 style to not specify namespace prefixes on child elements. Note that XML rules define that child elements  
1352 specified without a namespace prefix are from the namespace of the parent element, and not from the  
1353 default namespace of the XML document. Throughout the document, whitespace within XML element  
1354 values is used for readability. In practice, a service can accept and strip leading and trailing whitespace  
1355 within element values as if whitespace had not been used.

1356 Syntax definitions in this document use Augmented BNF (ABNF) as defined in IETF [RFC5234](#) with the  
1357 following exceptions:

- 1358 • Rules separated by a bar (|) represent choices, instead of using a forward slash (/) as defined in  
1359 ABNF.
- 1360 • Any characters must be processed case sensitively, instead of case-insensitively as defined in  
1361 ABNF.
- 1362 • Whitespace (i.e., the space character U+0020 and the tab character U+0009) is allowed between  
1363 syntactical elements, instead of assembling elements without whitespace as defined in ABNF.

1364

## ANNEX B (normative)

### OVF XSD

1365  
1366  
1367  
1368

1369 Normative copies of the XML Schemas for this specification may be retrieved by resolving the following  
1370 URLs:

1371  
1372  
1373

<http://schemas.dmtf.org/ovf/envelope/2/dsp8023.xsd>  
<http://schemas.dmtf.org/ovf/environment/1/dsp8027.xsd>

1374 Any `xs:documentation` content in XML Schemas for this specification is informative and provided only  
1375 for convenience.

1376 Normative copies of the XML Schemas for the WS-CIM mapping ([DSP0230](#)) of  
1377 `CIM_ResourceAllocationSystemSettingsData`, `CIM_VirtualSystemSettingData`,  
1378 `CIM_EthernetPortAllocationSettingData`, `CIM_StorageAllocationSettingData` and  
1379 `CIM_OperatingSystem`, may be retrieved by resolving the following URLs:

1380  
1381  
1382  
1383  
1384  
1385  
1386

[http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM\\_VirtualSystemSettingData.xsd](http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData.xsd)  
[http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM\\_ResourceAllocationSettingData.xsd](http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData.xsd)  
[http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM\\_EthernetPortAllocationSettingData.xsd](http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_EthernetPortAllocationSettingData.xsd)  
[http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM\\_StorageAllocationSettingData.xsd](http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_StorageAllocationSettingData.xsd)

1387 This specification is based on the following CIM MOFs:

1388  
1389  
1390  
1391  
1392  
1393

`CIM_VirtualSystemSettingData.mof`  
`CIM_ResourceAllocationSettingData.mof`  
`CIM_EthernetPortAllocationSettingData.mof`  
`CIM_StorageAllocationSettingData.mof`  
`CIM_OperatingSystem.mof`

## ANNEX C (informative)

### OVF mime type registration template

- 1394  
1395  
1396  
1397
- 1398 Registration Template
- 1399 To: ietf-types@iana.org
- 1400 Subject: Registration of media type Application/OVF
- 1401 Type name: Application
- 1402 Subtype name: OVF
- 1403 Required parameters: none
- 1404 Optional parameters: none
- 1405 Encoding considerations: binary
- 1406 Security considerations:
- 1407 • An OVF package contains active content that is expected to be launched in a virtual system.  
1408 The OVF standard, section 5.1 states: “An OVF package may be signed by signing the manifest  
1409 file. The digest of the manifest file is stored in a certificate file with extension .cert file along with  
1410 the base64-encoded X.509 certificate. The .cert file shall have the same base name as the .ovf  
1411 file and be a sibling of the .ovf file. A consumer of the OVF package shall verify the signature  
1412 and should validate the certificate.”
  - 1413 • An OVF package may contain passwords as part of the configuration information. The OVF  
1414 standard, section 9.5 states: “The optional Boolean attribute `ovf:password` indicates that the  
1415 property value may contain sensitive information. The default value is FALSE. OVF  
1416 implementations prompting for property values are advised to obscure these values when  
1417 `ovf:password` is set to TRUE. This is similar to HTML text input of type password. Note that  
1418 this mechanism affords limited security protection only. Although sensitive values are masked  
1419 from casual observers, default values in the OVF descriptor and assigned values in the OVF  
1420 environment are still passed in clear text.”
- 1421 Interoperability considerations:
- 1422 • OVF has demonstrated interoperability via multiple, interoperating implementations in the  
1423 market.
- 1424 Published specification:
- 1425 • DSP0243\_2.0.0.pdf
- 1426 Applications that use this media type:
- 1427 • Implementations of the DMTF Standard: Cloud Infrastructure Management Interface (CIM) (DSP0263\_1.0.0.pdf)
  - 1428
  - 1429 • Implementations of the SNIA Cloud Data Management Interface (CDMI) – OVF Extension
- 1430 Additional information:
- 1431 • Magic number(s): none

- 1432 • File extension(s): .ova
- 1433 • Macintosh file type code(s): none
- 1434 • Person & email address to contact for further information:
- 1435 • Intended usage: (One of COMMON, LIMITED USE or OBSOLETE.)
- 1436 • Restrictions on usage: (Any restrictions on where the media type can be used go here.)
- 1437 • Author:
- 1438 • Change controller:
- 1439

## ANNEX D (informative)

### OVF examples

1440  
1441  
1442  
1443

#### D.1 Examples of OVF package structure

1444  
1445  
1446  
1447  
1448  
1449  
1450  
1451  
1452  
1453

##### EXAMPLE 1:

The following list of files is an example of an OVF package:

```
package.ovf
package.mf
de-DE-resources.xml
vmdisk1.vmdk
vmdisk2.vhd
resource.iso
```

1454  
1455  
1456  
1457  
1458

##### EXAMPLE 2:

The following example show the partial contents of a manifest file:

```
SHA256(package.ovf)= 9902cc5ec4f4a00cabbff7b60d039263587ab430d5fbd5c5cd5e8707391c90a4
SHA256(vmdisk.vmdk)= aab66c4d70e17cec2236a651a3fc618cafc5ec6424122904dc0b9c286fce40c2
```

1459

##### EXAMPLE 3:

The following list of files is an example of a signed OVF package:

1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467

```
package.ovf
package.mf
package.cert
de-DE-resources.xml
vmdisk1.vmdk
vmdisk2.vmdk
resource.iso
```

1468

1469

##### EXAMPLE 4:

The following example shows the contents of a sample OVF certification file, where the SHA1 digest of the manifest file has been signed with a 512 bit key:

1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1480  
1481  
1482  
1483  
1484

```
SHA1(package.mf)= 7f4b8efb8fe20c06df1db68281a63f1b088e19dbf00e5af9db5e8e3e319de
7019db88a3bc699bab6ccd9e09171e21e88ee20b5255cec3fc28350613b2c529089
```

```
-----BEGIN CERTIFICATE-----
```

```
MIIBgjCCASwCAQQwDQYJKoZIhvcNAQEEBQAwwODELMAkGA1UEBhMCQVUxDDAKBgNV
BAgTA1FMRDEbMBkGA1UEAxMSU1NMZWF5L3JzYSB0ZXN0IENBMB4XDTEkMTAwOTIz
MzIwNVowXDTk4MDEwNTIzMzIwNVowYDELMAkGA1UEBhMCQVUxDDAKBgNVBAgTA1FM
RDEZMBcGA1UEChMQTWluY29tIFB0eS4gTHRkLjELMAkGA1UECzMxMxGzAZBgNV
BAMTElNTTGVheSBkZWlvIHN1cnZlcjBcMA0GCSqGSIb3DQEBAQUAA0sAMEgCQQC3
LCXcScWua0PFLkHBLm2VejqpA1F4RQ8q0VjRiPafjx/Z/aWH3ipdMVvuJGa/wFXb
/nDFLDlFwP+oCPwhBtVPAgMBAAEwDQYJKoZIhvcNAQEEBQADQQArNFsihWIjBzb0
DcsU0BvL2bvSwJrPEqFlkDq3F4M6EgutL9axEcANWgbbEdAvNJD1dmEmoWny27Pn
Ims6ZOZB
```

```
-----END CERTIFICATE-----
```

#### D.2 Examples of distribution of files

1485  
1486  
1487  
1488  
1489

##### EXAMPLE 1:

An example of an OVF package as a compressed archive:

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

1490 EXAMPLE 2:  
 1491 An example of an OVF package as a set of files on Web server follows:  
 1492 <http://mywebsite/virtualappliances/package.ovf>  
 1493 <http://mywebsite/virtualappliances/vmdisk1.vmdk>  
 1494 <http://mywebsite/virtualappliances/vmdisk2.vmdk>  
 1495 <http://mywebsite/virtualappliances/resource.iso>  
 1496 <http://mywebsite/virtualappliances/de-DE-resources.xml>

### 1497 D.3 Example of envelope element

1498 An example of the structure of an OVF descriptor with the top-level Envelope element  
 1499 follows:

```

1500 <?xml version="1.0" encoding="UTF-8"?>
1501 <Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1502 xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-
1503 schema/2/CIM_VirtualSystemSettingData"
1504 xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
1505 schema/2/CIM_ResourceAllocationSettingData"
1506 xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/2"
1507 xmlns="http://schemas.dmtf.org/ovf/envelope/2"
1508 xml:lang="en-US">
1509 <References>
1510 <File ovf:id="de-DE-resources.xml" ovf:size="15240"
1511 ovf:href="http://mywebsite/virtualappliances/de-DE-resources.xml"/>
1512 <File ovf:id="file1" ovf:href="vmdisk1.vmdk" ovf:size="180114671"/>
1513 <File ovf:id="file2" ovf:href="vmdisk2.vmdk" ovf:size="4882023564"
1514 ovf:chunkSize="2147483648"/>
1515 <File ovf:id="file3" ovf:href="resource.iso" ovf:size="212148764"
1516 ovf:compression="gzip"/>
1517 <File ovf:id="icon" ovf:href="icon.png" ovf:size="1360"/>
1518 </References>
1519 <!-- Describes meta-information about all virtual disks in the package -->
1520 <DiskSection>
1521 <Info>Describes the set of virtual disks</Info>
1522 <!-- Additional section content -->
1523 </DiskSection>
1524 <!-- Describes all networks used in the package -->
1525 <NetworkSection>
1526 <Info>List of logical networks used in the package</Info>
1527 <!-- Additional section content -->
1528 </NetworkSection>
1529 <SomeSection ovf:required="false">
1530 <Info>A plain-text description of the content</Info>
1531 <!-- Additional section content -->
1532 </SomeSection>
1533 <!-- Additional sections can follow -->
1534 <VirtualSystemCollection ovf:id="Some Product">
1535 <!-- Additional sections including VirtualSystem or VirtualSystemCollection-->
1536 </VirtualSystemCollection >
1537 <Strings xml:lang="de-DE">
1538 <!-- Specification of message resource bundles for de-DE locale -->
1539 </Strings>
1540 </Envelope>

```

## 1541 D.4 Example of file references

1542 EXAMPLE 1:

1543 The following example shows different types of file references:

```
1544 <File ovf:id="disk1" ovf:href="disk1.vmdk"/>
1545 <File ovf:id="disk2" ovf:href="disk2.vmdk" ovf:size="5368709120"
1546 ovf:chunkSize="2147483648"/>
1547 <File ovf:id="iso1" ovf:href="resources/image1.iso"/>
1548 <File ovf:id="iso2" ovf:href="http://mywebsite/resources/image2.iso"/>
```

1550 EXAMPLE 2:

1551 The following example shows manifest entries corresponding to the file references  
1552 above:

```
1553 SHA1(disk1.vmdk)= 3e19644ec2e806f38951789c76f43e4a0ec7e233
1554 SHA1(disk2.vmdk.000000000)= 4f7158731ff434380bf217da248d47a2478e79d8
1555 SHA1(disk2.vmdk.000000001)= 12849daeeaf43e7a89550384d26bd437bb8defaf
1556 SHA1(disk2.vmdk.000000002)= 4cdd21424bd9eeafa4c42112876217de2ee5556d
1557 SHA1(resources/image1.iso)= 72b37ff3fdd09f2a93f1b8395654649b6d06b5b3
1558 SHA1(http://mywebsite/resources/image2.iso)=
1559 d3c2d179011c970615c5cf10b30957d1c4c968ad
```

## 1560 D.5 Example of content element

1561 An example of a VirtualSystem element structure follows:

```
1562 <VirtualSystem ovf:id="simple-app">
1563 <Info>A virtual system</Info>
1564 <Name>Simple Appliance</Name>
1565 <SomeSection>
1566 <!-- Additional section content -->
1567 </SomeSection>
1568 <!-- Additional sections can follow -->
1569 </VirtualSystem>
```

1571 An example of a VirtualSystemCollection element structure follows:

```
1572 <VirtualSystemCollection ovf:id="multi-tier-app">
1573 <Info>A collection of virtual systems</Info>
1574 <Name>Multi-tiered Appliance</Name>
1575 <SomeSection>
1576 <!-- Additional section content -->
1577 </SomeSection>
1578 <!-- Additional sections can follow -->
1579 <VirtualSystem ovf:id="...">
1580 <!-- Additional sections -->
1581 </VirtualSystem>
1582 <!-- Additional VirtualSystem or VirtualSystemCollection elements can follow-->
1583 </VirtualSystemCollection>
```

## 1584 D.6 Examples of extensibility

1585 EXAMPLE 1:

```
1586 <!-- Optional custom section example -->
1587 <othersns:IncidentTrackingSection ovf:required="false">
1588 <Info>Specifies information useful for incident tracking purposes</Info>
1589 <BuildSystem>Acme Corporation Official Build System</BuildSystem>
1590 <BuildNumber>102876</BuildNumber>
1591 <BuildDate>10-10-2008</BuildDate>
1592 </othersns:IncidentTrackingSection>
```



```

1593
1594 EXAMPLE 2:
1595 <!-- Open content example (extension of existing type) -->
1596 <AnnotationSection>
1597 <Info>Specifies an annotation for this virtual system</Info>
1598 <Annotation>This is an example of how a future element (Author) can still be
1599 parsed by older clients</Annotation>
1600 <!-- AnnotationSection extended with Author element -->
1601 <others:Author ovf:required="false">John Smith</others:Author>
1602 </AnnotationSection>
1603
1604 EXAMPLE 3:
1605 <!-- Optional custom attribute example -->
1606 <Network ovf:name="VS network" others:desiredCapacity="1 Gbit/s">
1607 <Description>The main network for VSs</Description>
1608 </Network>

```

## 1609 D.7 Examples of VirtualHardwareSection

```

1610 EXAMPLE 1:
1611 Example of VirtualHardwareSection:
1612 <VirtualHardwareSection>
1613 <Info>Memory = 4 GB, CPU = 1 GHz, Disk = 100 GB, 1 Ethernet nic</Info>
1614 <Item>
1615 <rasd:AllocationUnits>Hertz*10^9</rasd:AllocationUnits>
1616 <rasd:Description>Virtual CPU</rasd:Description>
1617 <rasd:ElementName>1 GHz virtual CPU</rasd:ElementName>
1618 <rasd:InstanceID>1</rasd:InstanceID>
1619 <rasd:Reservation>1</rasd:Reservation>
1620 <rasd:ResourceType>3</rasd:ResourceType>
1621 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
1622 <rasd:VirtualQuantityUnit>Count</ rasd:VirtualQuantityUnit>
1623 </Item>
1624 <Item>
1625 <rasd:AllocationUnits>byte*2^30</rasd:AllocationUnits>
1626 <rasd:Description>Memory</rasd:Description>
1627 <rasd:ElementName>1 GByte of memory</rasd:ElementName>
1628 <rasd:InstanceID>2</rasd:InstanceID>
1629 <rasd:Limit>4</rasd:Limit>
1630 <rasd:Reservation>4</rasd:Reservation>
1631 <rasd:ResourceType>4</rasd:ResourceType>
1632 </Item>
1633 <EthernetPortItem>
1634 <rasd:AllocationUnits>bit / second *2^30 </rasd:AllocationUnits>
1635 <epasd:Connection>VS Network</epasd:Connection>
1636 <epasd:Description>Virtual NIC</epasd:Description>
1637 <epasd:ElementName>Ethernet Port</epasd:ElementName>
1638 <epasd:NetworkPortProfileID>1</epasd:NetworkPortProfileID>
1639 <epasd:NetworkPortProfileIDType>4</epasd:NetworkPortProfileIDType>
1640 <epasd:ResourceType>10</epasd:ResourceType>
1641 <epasd:VirtualQuantity>1</epasd:VirtualQuantity>
1642 <epasd:VirtualQuantityUnits>Count</epasd:VirtualQuantityUnits>
1643 </EthernetPortItem>
1644 <StorageItem>
1645 <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>

```

```

1646 <sasd:Description>Virtual Disk</sasd:Description>
1647 <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
1648 <sasd:Reservation>100</sasd:Reservation>
1649 <sasd:ResourceType>31</sasd:ResourceType>
1650 <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
1651 <sasd:VirtualQuantityUnit>Count</sasd:VirtualQuantityUnit>
1652 </StorageItem>
1653 </VirtualHardwareSection>
1654
1655 EXAMPLE 2:
1656 <rasd:ResourceSubType>buslogic lsilogic</rasd:ResourceSubType>

```

## 1657 D.8 Examples of virtual hardware elements

```

1658 EXAMPLE 1:
1659 The following example shows a description of memory size:
1660 <Item>
1661 <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1662 <rasd:Description>Memory Size</rasd:Description>
1663 <rasd:ElementName>256 MB of memory</rasd:ElementName>
1664 <rasd:InstanceID>2</rasd:InstanceID>
1665 <rasd:ResourceType>4</rasd:ResourceType>
1666 <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
1667 </Item>
1668
1669 EXAMPLE 2:
1670 The following example shows a description of a virtual Ethernet adapter:
1671 <EthernetPortItem>
1672 <epasd:Address>00-16-8B-DB-00-5E</epasd:Address>
1673 <epasd:Connection>VS Network</epasd:Connection>
1674 <epasd:Description>Virtual NIC</epasd:Description>
1675
1676 <epasd:ElementName>Ethernet Port 1</epasd:ElementName>
1677 <epasd:InstanceID>3</epasd:InstanceID>
1678 <epasd:NetworkPortProfileID>1</epasd:NetworkPortProfileID>
1679 <epasd:NetworkPortProfileIDType>4</epasd:NetworkPortProfileIDType>
1680 <epasd:VirtualQuantityUnits>1</epasd:VirtualQuantityUnits>
1681 </EthernetPortItem>
1682
1683 EXAMPLE 3:
1684 The following example shows a description of a virtual storage:
1685 <StorageItem>
1686 <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
1687 <sasd:Description>Virtual Disk</sasd:Description>
1688 <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
1689 <sasd:InstanceID>4</sasd:InstanceID>
1690 <sasd:Reservation>100</sasd:Reservation>
1691 <sasd:ResourceType>31</sasd:ResourceType>
1692 <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
1693 </StorageItem>

```

## 1694 D.9 Example of ranges on elements

```

1695 EXAMPLE:
1696 The following example shows the use of range markers:
1697 <VirtualHardwareSection>

```

```

1698 <Info>...</Info>
1699 <Item>
1700 <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1701 <rasd:ElementName>512 MB memory size</rasd:ElementName>
1702 <rasd:InstanceID>0</rasd:InstanceID>
1703 <rasd:ResourceType>4</rasd:ResourceType>
1704 <rasd:VirtualQuantity>512</rasd:VirtualQuantity>
1705 </Item>
1706 <Item ovf:bound="min">
1707 <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1708 <rasd:ElementName>384 MB minimum memory size</rasd:ElementName>
1709 <rasd:InstanceID>0</rasd:InstanceID>
1710 <rasd:Reservation>384</rasd:Reservation>
1711 <rasd:ResourceType>4</rasd:ResourceType>
1712 </Item>
1713 <Item ovf:bound="max">
1714 <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1715 <rasd:ElementName>1024 MB maximum memory size</rasd:ElementName>
1716 <rasd:InstanceID>0</rasd:InstanceID>
1717 <rasd:Reservation>1024</rasd:Reservation>
1718 <rasd:ResourceType>4</rasd:ResourceType>
1719 </Item>
1720 </VirtualHardwareSection>

```

## 1721 D.10 Example of DiskSection

1722 EXAMPLE: The following example shows a description of virtual disks:

```

1723 <DiskSection>
1724 <Info>Describes the set of virtual disks</Info>
1725 <Disk ovf:diskId="vmdisk1" ovf:fileRef="file1" ovf:capacity="8589934592"
1726 ovf:populatedSize="3549324972"
1727 ovf:format=
1728 "http://www.vmware.com/interfaces/specifications/vmdk.html#sparse">
1729 </Disk>
1730 <Disk ovf:diskId="vmdisk2" ovf:capacity="536870912">
1731 </Disk>
1732 <Disk ovf:diskId="vmdisk3" ovf:capacity="${disk.size}"
1733 ovf:capacityAllocationUnits="byte * 2^30">
1734 </Disk>
1735 </DiskSection>

```

## 1736 D.11 Example of NetworkSection

```

1737 <NetworkSection>
1738 <Info>List of logical networks used in the package</Info>
1739 <Network ovf:name="VS Network">
1740 <Description>The network that the service will be available on</Description>
1741 <NetworkPortProfile>
1742 <Item>
1743 <epasd:AllocationUnits>GigaBits per Second</epasd:AllocationUnits>
1744 <epasd:ElementName>Network Port Profile 1</epasd:ElementName>
1745 <epasd:InstanceID>1</epasd:InstanceID>
1746 <epasd:NetworkPortProfileID>1</epasd:NetworkPortProfileID>
1747 <epasd:NetworkPortProfileIDType>4</epasd:NetworkPortProfileIDType>
1748 <epasd:Reservation>1</epasd:Reservation>
1749 </Item>

```

```

1750 </NetworkPortProfile>
1751 </Network>
1752 </NetworkSection>

```

## 1753 D.12 Example of ResourceAllocationSection

```

1754 <ResourceAllocationSection>
1755 <Info>Defines reservations for CPU and memory for the collection of VSs</Info>
1756 <Item>
1757 <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1758 <rasd:ElementName>300 MB reservation</rasd:ElementName>
1759 <rasd:InstanceID>0</rasd:InstanceID>
1760 <rasd:Reservation>300</rasd:Reservation>
1761 <rasd:ResourceType>4</rasd:ResourceType>
1762 </Item>
1763 <Item ovf:configuration="..." ovf:bound="...">
1764 <rasd:AllocationUnits>hertz * 10^6</rasd:AllocationUnits>
1765 <rasd:ElementName>500 MHz reservation</rasd:ElementName>
1766 <rasd:InstanceID>0</rasd:InstanceID>
1767 <rasd:Reservation>500</rasd:Reservation>
1768 <rasd:ResourceType>3</rasd:ResourceType>
1769 </Item>
1770 <EthernetPortItem>
1771 <epasd:Address>00-16-8B-DB-00-5E</epasd:Address>
1772 <epasd:Connection>VS Network</epasd:Connection>
1773 <epasd:Description>Virtual NIC</epasd:Description>
1774 <epasd:ElementName>Ethernet Port 1</epasd:ElementName>
1775 <epasd:InstanceID>3</epasd:InstanceID>
1776 <epasd:NetworkPortProfileID>1</epasd:NetworkPortProfileID>
1777 <epasd:NetworkPortProfileIDType>4</epasd:NetworkPortProfileIDType>
1778 <epasd:VirtualQuantityUnits>1</epasd:VirtualQuantityUnits>
1779 </EthernetPortItem>
1780 <StorageItem>
1781 <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
1782 <sasd:Description>Virtual Disk</sasd:Description>
1783 <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
1784 <sasd:InstanceID>4</sasd:InstanceID>
1785 <sasd:Reservation>100</sasd:Reservation>
1786 <sasd:ResourceType>31</sasd:ResourceType>
1787 <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
1788 </StorageItem>
1789 </ResourceAllocationSection>

```

## 1790 D.13 Example of annotation

```

1791 <AnnotationSection>
1792 <Info>An annotation on this service. It can be ignored</Info>
1793 <Annotation>Contact customer support if you have any problems</Annotation>
1794 </AnnotationSection >

```

## 1795 D.14 Example of Product section

```

1796 <ProductSection ovf:class="com.mycrm.myservice" ovf:instance="1">
1797 <Info>Describes product information for the service</Info>
1798 <Product>MyCRM Enterprise</Product>
1799 <Vendor>MyCRM Corporation</Vendor>
1800 <Version>4.5</Version>

```

```

1801 <FullVersion>4.5-b4523</FullVersion>
1802 <ProductUrl>http://www.mycrm.com/enterprise</ProductUrl>
1803 <VendorUrl>http://www.mycrm.com</VendorUrl>
1804 <Icon ovf:height="32" ovf:width="32" ovf:mimeType="image/png" ovf:fileRef="icon">
1805 <Category>Email properties</Category>
1806 <Property ovf:key="adminemail" ovf:type="string" ovf:userConfigurable="true">
1807 <Label>Admin email</Label>
1808 <Description>Email address of administrator</Description>
1809 </Property>
1810 <Category>Admin properties</Category>
1811 <Property ovf:key="app_log" ovf:type="string" ovf:value="low"
1812 ovf:userConfigurable="true">
1813 <Description>Loglevel for the service</Description>
1814 </Property>
1815 <Property ovf:key="app_isSecondary" ovf:value="false" ovf:type="boolean">
1816 <Description>Cluster setup for application server</Description>
1817 </Property>
1818 <Property ovf:key="app_ip" ovf:type="string" ovf:value="${appserver-vm}">
1819 <Description>IP address of the application server VS</Description>
1820 </Property>
1821 </ProductSection>

```

## 1822 D.15 Example of EULA section

```

1823 <EulaSection>
1824 <Info>Licensing agreement</Info>
1825 <License>
1826 Lorem ipsum dolor sit amet, ligula suspendisse nulla pretium, rhoncus tempor placerat
1827 fermentum, enim integer ad vestibulum volutpat. Nisl rhoncus turpis est, vel elit,
1828 congue wisi enim nunc ultricies sit, magna tincidunt. Maecenas aliquam maecenas ligula
1829 nostra, accumsan taciti. Sociis mauris in integer, a dolor netus non dui aliquet,
1830 sagittis felis sodales, dolor sociis mauris, vel eu libero cras. Interdum at. Eget
1831 habitasse elementum est, ipsum purus pede porttitor class, ut adipiscing, aliquet sed
1832 auctor, imperdiet arcu per diam dapibus libero duis. Enim eros in vel, volutpat nec
1833 pellentesque leo, scelerisque.
1834 </License>
1835 </EulaSection>

```

## 1836 D.16 Example of StartupSection

```

1837 <StartupSection>
1838 <Item ovf:id="vm1" ovf:order="0" ovf:startDelay="30" ovf:stopDelay="0"
1839 ovf:startAction="powerOn" ovf:waitingForGuest="true"
1840 ovf:stopAction="powerOff"/>
1841 <Item ovf:id="teamA" ovf:order="0"/>
1842 <Item ovf:id="vm2" ovf:order="1" ovf:startDelay="0" ovf:stopDelay="20"
1843 ovf:startAction="powerOn" ovf:stopAction="guestShutdown"/>
1844 </StartupSection>

```

## 1845 D.17 Example of DeploymentOptionSection

```

1846 <DeploymentOptionSection>
1847 <Configuration ovf:id="minimal">
1848 <Label>Minimal</Label>
1849 <Description>Some description</Description>
1850 </Configuration>
1851 <Configuration ovf:id="normal" ovf:default="true">

```

```

1852 <Label>Typical</Label>
1853 <Description>Some description</Description>
1854 </Configuration>
1855 <!-- Additional configurations -->
1856 </DeploymentOptionSection>
1857
1858 EXAMPLE 1: The following example shows a VirtualHardwareSection:
1859 <VirtualHardwareSection>
1860 <Info>...</Info>
1861 <Item>
1862 <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1863 <rasd:ElementName>512 MB memory size and 256 MB reservation</rasd:ElementName>
1864 <rasd:InstanceID>0</rasd:InstanceID>
1865 <rasd:Reservation>256</rasd:Reservation>
1866 <rasd:ResourceType>4</rasd:ResourceType>
1867 <rasd:VirtualQuantity>512</rasd:VirtualQuantity>
1868 </Item>
1869 ...
1870 <Item ovf:configuration="big">
1871 <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1872 <rasd:ElementName>1024 MB memory size and 512 MB reservation</rasd:ElementName>
1873 <rasd:InstanceID>0</rasd:InstanceID>
1874 <rasd:Reservation>512</rasd:Reservation>
1875 <rasd:ResourceType>4</rasd:ResourceType>
1876 <rasd:VirtualQuantity>1024</rasd:VirtualQuantity>
1877 </Item>
1878 </VirtualHardwareSection>
1879
1880 EXAMPLE 2: The following shows an example ProductSection:
1881 <ProductSection>
1882 <Property ovf:key="app_adminEmail" ovf:type="string" ovf:userConfigurable="true"
1883 ovf:configuration="standard">
1884 <Label>Admin email</Label>
1885 <Description>Email address of service administrator</Description>
1886 </Property>
1887 <Property ovf:key="app_log" ovf:type="string" ovf:value="low"
1888 ovf:userConfigurable="true">
1889 <Label>Loglevel</Label>
1890 <Description>Loglevel for the service</Description>
1891 <Value ovf:value="none" ovf:configuration="minimal">
1892 </Property>
1893 </ProductSection>
1894 In the example above, the app_adminEmail property is only user configurable in the
1895 standard configuration, while the default value for the app_log property is changed
1896 from low to none in the minimal configuration.

```

## 1897 D.18 Example of OperatingSystemSection

```

1898 <OperatingSystemSection ovf:id="76">
1899 <Info>Specifies the operating system installed</Info>
1900 <Description>Microsoft Windows Server 2008</Description>
1901 </OperatingSystemSection>

```

## 1902 D.19 Example of InstallSection

```

1903 <InstallSection ovf:initialBootStopDelay="300">

```

```

1904 <Info>Specifies that the virtual system needs to be booted once after having
1905 created the guest software in order to install and/or configure the software
1906 </Info>
1907 </InstallSection>

```

## 1908 **D.20 Example of EnvironmentFilesSection**

1909 EXAMPLE:

```

1910 <Envelope>
1911 <References>
1912 ...
1913 <File ovf:id="config" ovf:href="config.xml" ovf:size="4332"/>
1914 <File ovf:id="resources" ovf:href="http://mywebsite/resources/resources.zip"/>
1915 </References>
1916 ...
1917 <VirtualSystem ovf:id="...">
1918 ...
1919 <ovf:EnvironmentFilesSection ovf:required="false" ovf:transport="iso">
1920 <Info>Config files to be included in OVF environment</Info>
1921 <ovf:File ovf:fileRef="config" ovf:path="setup/cfg.xml"/>
1922 <ovf:File ovf:fileRef="resources" ovf:path="setup/resources.zip"/>
1923 </ovf:EnvironmentFilesSection>
1924 ...
1925 </VirtualSystem>
1926 ...
1927 </Envelope>

```

1928 In the example above, the file config.xml in the OVF package will be copied to the OVF  
1929 environment ISO image and be accessible to the guest software in location  
1930 /ovffiles/setup/cfg.xml, while the file resources.zip will be accessible in location  
1931 /ovffiles/setup/resources.zip.

## 1932 **D.21 Example of BootDeviceSection**

1933 In the example below, the Pre-Install configuration specifies the boot source as a  
1934 specific device (network), while the Post-Install configuration specifies a device  
1935 type (hard disk).

1936 EXAMPLE:

```

1937 <Envelope>
1938 ...
1939 <VirtualSystem ovf:id="...">
1940 ...
1941 <ovf:BootDeviceSection>
1942 <Info>Boot device order specification</Info>
1943 <bootc:CIM_BootConfigSetting>
1944 <bootc:Caption>Pre-Install</bootc:Caption>
1945 <bootc:Description>Boot Sequence for fixup of disk</bootc:Description>
1946 <boots:CIM_BootSourceSetting>
1947 <boots:Caption>Fix-up DVD on the network</boots:Caption>
1948 <boots:InstanceID>3</boots:InstanceID> <!-- Network device-->
1949 </boots:CIM_BootSourceSetting>
1950 <boots:CIM_BootSourceSetting>
1951 <boots:Caption>Boot virtual disk</boots:Caption>
1952 <boots:StructuredBootString>CIM:Hard-Disk</boots:StructuredBootString>
1953 </boots:CIM_BootSourceSetting>
1954 </bootc:CIM_BootConfigSetting>
1955 </ovf:BootDeviceSection>

```

```

1956 ...
1957 </VirtualSystem>
1958 </Envelope>

```

## 1959 D.22 Example of SharedDiskSection

```

1960 EXAMPLE:
1961 <ovf:SharedDiskSection>
1962 <Info>Describes the set of virtual disks shared between VSs</Info>
1963 <ovf:SharedDisk ovf:diskId="datadisk" ovf:fileRef="data"
1964 ovf:capacity="8589934592" ovf:populatedSize="3549324972"
1965 ovf:format=
1966 "http://www.vmware.com/interfaces/specifications/vmdk.html#sparse"/>
1967 <ovf:SharedDisk ovf:diskId="transientdisk" ovf:capacity="536870912"/>
1968 </ovf:SharedDiskSection>

```

## 1969 D.23 Example of ScaleOutSection

```

1970 EXAMPLE:
1971 <VirtualSystemCollection ovf:id="web-tier">
1972 ...
1973 <ovf:ScaleOutSection ovf:id="web-server">
1974 <Info>Web tier</Info>
1975 <ovf:Description>Number of web server instances in web tier</ovf:Description>
1976 <ovf:InstanceCount ovf:default="4" ovf:minimum="2" ovf:maximum="8"/>
1977 </ovf:ScaleOutSection>
1978 ...
1979 <VirtualSystem ovf:id="web-server">
1980 <Info>Prototype web server</Info>
1981 ...
1982 </VirtualSystem>
1983 </VirtualSystemCollection>

```

1985 In the example above, the deployment platform creates a web tier containing between  
1986 two and eight web server virtual system instances, with a default instance count of  
1987 four. The deployment platform makes an appropriate choice (e.g., by prompting the  
1988 user). Assuming three replicas were created, the OVF environment available to the  
1989 guest software in the first replica has the following content structure:

```

1990 EXAMPLE:
1991 <Environment ... ovfenv:id="web-server-1">
1992 ...
1993 <Entity ovfenv:id="web-server-2">
1994 ...
1995 </Entity>
1996 <Entity ovfenv:id="web-server-3">
1997 ...
1998 </Entity>
1999 </Environment>

```

```

2000 EXAMPLE:
2001 <VirtualSystemCollection ovf:id="web-tier">
2002 ...
2003 <DeploymentOptionSection>
2004 <Info>Deployment size options</Info>
2005 <Configuration ovf:id="minimal">

```



```

2008 <Label>Minimal</Label>
2009 <Description>Minimal deployment scenario</Description>
2010 </Configuration>
2011 <Configuration ovf:id="common" ovf:default="true">
2012 <Label>Typical</Label>
2013 <Description>Common deployment scenario</Description>
2014 </Configuration>
2015 ...
2016 </DeploymentOptionSection>
2017 ...
2018 <ovf:ScaleOutSection ovf:id="web-server">
2019 <Info>Web tier</Info>
2020 <ovf:Description>Number of web server instances in web tier</ovf:Description>
2021 <ovf:InstanceCount ovf:default="4"/>
2022 <ovf:InstanceCount ovf:default="1" ovf:configuration="minimal"/>
2023 </ovf:ScaleOutSection>
2024 ...
2025 </VirtualSystemCollection>

```

2026 In the example above, the default replica count is four, unless the minimal deployment  
 2027 scenario is chosen, in which case the default is one.

## 2028 D.24 Example of PlacementGroupSection

2029 EXAMPLE:

```

2030 <Envelope ...>
2031 ...
2032 <ovf:PlacementGroupSection ovf:id="web" ovf:policy="availability">
2033 <Info>Placement policy for group of VSs</Info>
2034 <ovf:Description>Placement policy for web tier</ovf:Description>
2035 </ovf:PlacementGroupSection>
2036 ...
2037 <VirtualSystemCollection ovf:id="web-tier">
2038 ...
2039 <ovf:ScaleOutSection ovf:id="web-node">
2040 <Info>Web tier</Info>
2041 ...
2042 </ovf:ScaleOutSection>
2043 ...
2044 <VirtualSystem ovf:id="web-node">
2045 <Info>Web server</Info>
2046 ...
2047 <ovf:PlacementSection ovf:group="web">
2048 <Info>Placement policy group reference</Info>
2049 </ovf:PlacementSection>
2050 ...
2051 </VirtualSystem>
2052 </VirtualSystemCollection>
2053 </Envelope>

```

2054 In the example above, all virtual systems in the compute tier should be placed  
 2055 separately for high availability. This example also use the ScaleOutSection defined in  
 2056 clause 9.14, in which case each replica get the policy assigned.

## 2057 D.25 Example of EncryptionSection

2058 Below is an example of an OVF encryption section with encryption methods utilized in  
 2059 the OVF document, and the corresponding reference list pointing to the items that have  
 2060 been encrypted.

```

2061
2062 EXAMPLE:
2063 <ovf:EncryptionSection>
2064 <!--- This section contains two different methods of encryption and the corresponding
2065 back pointers to the data that is encrypted -->
2066 <!--- Method#1: Pass phrase based Key derivation -->
2067 <!--- The following derived key block defines PBKDF2 and the corresponding back
2068 pointers to the encrypted data elements -->
2069 <!--- Use a salt value "ovfpassword" and iteration count of 4096 --->
2070 <xenc11:DerivedKey>
2071 <xenc11:KeyDerivationMethod
2072 Algorithm="http://www.rsasecurity.com/rsalabs/pkcs/schemas/pkcs-5#pbkdf2"/>
2073 <pkcs-5:PBKDF2-params>
2074 <Salt>
2075 <Specified>ovfpassword</Specified>
2076 </Salt>
2077 <IterationCount>4096</IterationCount>
2078 <KeyLength>16</KeyLength>
2079 <PRF Algorithm="http://www.w3.org/2001/04/xmldsig-more#hmac-
2080 sha256"/>
2081 </pkcs-5:PBKDF2-params>
2082 ...
2083 <!-- The ReferenceList element below contains references to the file Ref-109.vhd via
2084 the URI syntax which is specified by XML Encryption.
2085 --->
2086 <xenc:ReferenceList>
2087 <xenc:DataReference URI="#first.vhd" />
2088 <xenc:DataReference URI=... />
2089 <xenc:DataReference URI=... />
2090 </xenc:ReferenceList>
2091 </xenc11:DerivedKey>
2092 <!-- Method#2: The following example illustrates use of a symmetric key
2093 transported using the public key within a certificate -->
2094 <xenc:EncryptedKey>
2095 <xenc:EncryptionMethod
2096 Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
2097 <ds:KeyInfo xmlns:ds='http://www.w3.org/2000/09/xmldsig#'
2098 <ds:X509Data>
2099 <ds:X509Certificate> ... </ds:X509Certificate>
2100 </ds:X509Data>
2101 </ds:KeyInfo>
2102 <xenc:CipherData>
2103 <xenc:CipherValue> ... </xenc:CipherValue>
2104 </xenc:CipherData>
2105 <!-- The ReferenceList element below contains reference #second-xml-fragment" to the
2106 XML fragment that has been encrypted using the above method --->
2107 <xenc:ReferenceList>
2108 <xenc:DataReference URI='#second-xml-fragment' />
2109 <xenc:DataReference URI='...' />
2110 <xenc:DataReference URI='...' />

```

```

2111 </xenc:ReferenceList>
2112 </xenc:EncryptedKey>
2113 </ovf:EncryptionSection>
2114 Below is an example of the encrypted file which is referenced in the EncryptionSection
2115 above using URI='Ref-109.vhd' syntax.
2116 EXAMPLE:
2117 <ovf:References>
2118 <ovf:File ovf:id="Xen:9cb10691-4012-4aeb-970c-3d47a906bfff/0b13bdba-3761-8622-22fc-
2119 2e252ed9ce14" ovf:href="Ref-109.vhd">
2120 <!-- the encrypted file referenced by the package is enclosed by an EncryptedData with
2121 a CipherReference to the actual encrypted file. The EncryptionSection in this example
2122 has a back pointer to it under the PBKDF2 algorithm via Id="first.vhd". This tells the
2123 decrypter how to decrypt the file -->
2124 <xenc:EncryptedData Id="first.vhd" Type='http://www.w3.org/2001/04/xmlenc#Element' >
2125 <xenc:EncryptionMethod
2126 Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc" />
2127 <xenc:CipherData>
2128 <xenc:CipherReference URI='Ref-109.vhd' />
2129 </xenc:CipherData>
2130 </xenc:EncryptedData>
2131 </ovf:File>
2132 </ovf:References>
2133 Below is an example of the encrypted OVF markup which is referenced in the
2134 EncryptionSection above using URI='#second-xml-fragment' syntax.
2135 EXAMPLE:
2136 <!-- the EncryptedData element below encompasses encrypted xml from the original
2137 document. It is provided with the Id "first-xml-fragment" which allows it to be
2138 referenced from the EncryptionSection. -->
2139 <xenc:EncryptedData Type=http://www.w3.org/2001/04/xmlenc#Element Id="second-xml-
2140 fragment">
2141 <!-- Each EncryptedData specifies its own encryption method. -->
2142 <xenc:EncryptionMethod Algorithm=http://www.w3.org/2001/04-xmlenc#aes128-cbc/>
2143 <xenc:CipherData>
2144 <!-- Encrypted content --->
2145 <xenc:CipherValue>DEADBEEF</xenc:CipherValue>
2146 </xenc:CipherData>
2147 </xenc:EncryptedData>

```

## 2148 **D.26 Example of internationalization**

```

2149 EXAMPLE 1:
2150 <Info ovf:msgid="info.text">Default info.text value if no locale is set or no locale
2151 match</Info>
2152 <License ovf:msgid="license.tomcat-6_0"/> <!-- No default message -->
2153
2154 Using Internal Resource Bundles
2155
2156 EXAMPLE 2:
2157 <ovf:Envelope xml:lang="en-US">
2158 ...
2159 ... sections and content here ...
2160 ...
2161 <Info msgid="info.os">Operating System</Info>
2162 ...
2163 <Strings xml:lang="da-DA">

```

```

2164 <Msg ovf:msgid="info.os">Operating System</Msg>
2165 ...
2166 </Strings>
2167 <Strings xml:lang="de-DE">
2168 <Msg ovf:msgid="info.os">Betriebssystem</Msg>
2169 ...
2170 </Strings>
2171 </ovf:Envelope>
2172

```

## 10.2 External Resource Bundles

### EXAMPLE 3:

```

2175 <ovf:Envelope xml:lang="en-US">
2176 <References>
2177 ...
2178 <File ovf:id="it-it-resources" ovf:href="resources/it-it-bundle.msg"/>
2179 </References>
2180 ... sections and content here ...
2181 ...
2182 <Strings xml:lang="it-IT" ovf:fileRef="it-it-resources"/>
2183 ...
2184 </ovf:Envelope>

```

EXAMPLE 4: Example content of external resources/it-it-bundle.msg file, which is referenced in previous example:

```

2187 <Strings
2188 xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1"
2189 xmlns="http://schemas.dmtf.org/ovf/envelope/1"
2190 xml:lang="it-IT">
2191 <Msg ovf:msgid="info.os">Sistema operativo</Msg>
2192 ...
2193 </Strings>

```

## D.27 Example of message content in an external file

### EXAMPLE:

```

2196 <Envelope xml:lang="en-US">
2197 <References>
2198 <File ovf:id="license-en-US" ovf:href="license-en-US.txt"/>
2199 <File ovf:id="license-de-DE" ovf:href="license-de-DE.txt"/>
2200 </References>
2201 ...
2202 <VirtualSystem ovf:id="...">
2203 <EulaSection>
2204 <Info>Licensing agreement</Info>
2205 <License ovf:msgid="license">Unused</License>
2206 </EulaSection>
2207 ...
2208 </VirtualSystem>
2209 ...
2210 <Strings xml:lang="en-US">
2211 <Msg ovf:msgid="license" ovf:fileRef="license-en-US">Invalid license</Msg>
2212 </Strings>
2213 <Strings xml:lang="de-DE">
2214 <Msg ovf:msgid="license" ovf:fileRef="license-de-DE">Ihre Lizenz ist nicht
2215 gültig</Msg>
2216 </Strings>

```

2217 </Envelope>  
 2218 In the example above, the default license agreement is stored in plain text file  
 2219 license-en-US.txt, while the license agreement for the de-DE locale is stored in file  
 2220 license-de-DE.txt.  
 2221 Note that the above mechanism works for all localizable elements and not just License.

## 2222 **D.28 Example of environment document**

2223 EXAMPLE: An example of the structure of the OVF environment document follows:  
 2224 <?xml version="1.0" encoding="UTF-8"?>  
 2225 <Environment xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 2226           xmlns:ovfenv="http://schemas.dmtf.org/ovf/environment/1"  
 2227           xmlns="http://schemas.dmtf.org/ovf/environment/1"  
 2228           ovfenv:id="identification of VS from OVF descriptor">  
 2229     <!-- Information about virtualization platform -->  
 2230     <PlatformSection>  
 2231       <Kind>Type of virtualization platform</Kind>  
 2232       <Version>Version of virtualization platform</Version>  
 2233       <Vendor>Vendor of virtualization platform</Vendor>  
 2234       <Locale>Language and country code</Locale>  
 2235       <TimeZone>Current timezone offset in minutes from UTC</TimeZone>  
 2236     </PlatformSection>  
 2237     <!-- Properties defined for this virtual system -->  
 2238     <PropertySection>  
 2239       <Property ovfenv:key="key" ovfenv:value="value">  
 2240        <!-- More properties -->  
 2241       </Property>  
 2242     <Entity ovfenv:id="id of sibling virtual system or virtual system collection">  
 2243       <PropertySection>  
 2244        <!-- Properties from sibling -->  
 2245        </PropertySection>  
 2246     </Entity>  
 2247 </Environment>

## ANNEX E (informative)

### Network port profile examples

#### E.1 Example 1 (OVF descriptor for one virtual system and one network with an inlined network port profile)

The example below shows an OVF descriptor that describes a virtual system and a network to which it connects. The virtual system description in this example uses an inlined network port profile that is described as an XML element that contains child XML elements from epasd namespace. The network described in the network section uses the same network port profile description. The network port profile described in this example is used to reserve 1 Gbps of bandwidth.

```

2259 <?xml version="1.0" encoding="UTF-8"?>
2260 <Envelope xsi:schemaLocation="http://schemas.dmtf.org/ovf/envelope/2
2261 file:///C:/dsp8023_2.0.0_wgv0.9.5.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2262 xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/2" xmlns="http://schemas.dmtf.org/ovf/envelope/2"
2263 xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
2264 xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
2265 xmlns:epasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2266 schema/2/CIM_EthernetPortAllocationSettingData"
2267 xmlns:sasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_StorageAllocationSettingData">
2268 <!-- References to all external files -->
2269 <References>
2270 <File ovf:id="file1" ovf:href="vmdisk1.vmdk" ovf:size="2000000000"/>
2271 </References>
2272 <!-- Describes meta-information for all virtual disks in the package -->
2273 <DiskSection>
2274 <Info>Describes the set of virtual disks</Info>
2275 <Disk ovf:diskId="vmdisk1" ovf:fileRef="file1" ovf:capacity="4294967296"
2276 ovf:format="http://www.examplecompany.com/interfaces/specifications/vmdk.html#sparse"/>
2277 </DiskSection>
2278 <!-- Describes all networks used in the package -->
2279 <NetworkSection>
2280 <Info>List of logical networks used in the package</Info>
2281 <Network ovf:name="VS Network">
2282 <Description>The network that the VSs connect to</Description>
2283 <NetworkPortProfile>
2284 <!-- Network port profile describing bandwidth reservation. Network port profile
2285 is identified by UUID. -->
2286 <Item>
2287 <epasd:AllocationUnits>bit / second * 10^9</epasd:AllocationUnits>
2288 <epasd:ElementName>Network Port Profile 1</epasd:ElementName>
2289 <epasd:InstanceID>1</epasd:InstanceID>
2290 <epasd:NetworkPortProfileID>aaaaaaaa-bbbb-cccc-dddd-
2291 eeeeeeeeeee</epasd:NetworkPortProfileID>
2292 <epasd:NetworkPortProfileIDType>3</epasd:NetworkPortProfileIDType>
2293 <epasd:Reservation>1</epasd:Reservation>
2294 </Item>
2295 </NetworkPortProfile>
2296 </Network>
2297 </NetworkSection>
2298 <VirtualSystem ovf:id="vm">
2299 <Info>Describes a virtual system</Info>
2300 <Name>Virtual Appliance One</Name>
2301 <ProductSection>
2302 <Info>Describes product information for the appliance</Info>
2303 <Product>The Great Appliance</Product>
2304 <Vendor>Some Great Corporation</Vendor>
2305 <Version>13.00</Version>
2306 <FullVersion>13.00-b5</FullVersion>
2307 <ProductUrl>http://www.somegreatcorporation.com/greatappliance</ProductUrl>
2308 <VendorUrl>http://www.somegreatcorporation.com/</VendorUrl>

```

```

2309 <Property ovf:key="adminemail" ovf:type="string">
2310 <Description>Email address of administrator</Description>
2311 </Property>
2312 <Property ovf:key="app_ip" ovf:type="string" ovf:defaultValue="192.168.0.10">
2313 <Description>The IP address of this appliance</Description>
2314 </Property>
2315 </ProductSection>
2316 <AnnotationSection ovf:required="false">
2317 <Info>A random annotation on this service. It can be ignored</Info>
2318 <Annotation>Contact customer support if you have any problems</Annotation>
2319 </AnnotationSection>
2320 <EulaSection>
2321 <Info>License information for the appliance</Info>
2322 <License>Insert your favorite license here</License>
2323 </EulaSection>
2324 <VirtualHardwareSection>
2325 <Info>Memory = 4 GB, CPU = 1 GHz, Disk = 100 GB, 1 Ethernet nic</Info>
2326 <Item>
2327 <rasd:AllocationUnits>Hertz*10^9</rasd:AllocationUnits>
2328 <rasd:Description>Virtual CPU</rasd:Description>
2329 <rasd:ElementName>1 GHz virtual CPU</rasd:ElementName>
2330 <rasd:InstanceID>1</rasd:InstanceID>
2331 <rasd:Reservation>1</rasd:Reservation>
2332 <rasd:ResourceType>3</rasd:ResourceType>
2333 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2334 </Item>
2335 <Item>
2336 <rasd:AllocationUnits>byte*2^30</rasd:AllocationUnits>
2337 <rasd:Description>Memory</rasd:Description>
2338 <rasd:ElementName>1 GByte of memory</rasd:ElementName>
2339 <rasd:InstanceID>2</rasd:InstanceID>
2340 <rasd:ResourceType>4</rasd:ResourceType>
2341 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2342 </Item>
2343 <EthernetPortItem>
2344 <epasd:Address>00-16-8B-DB-00-5E</epasd:Address>
2345 <epasd:AllocationUnits>bit / second * 10^9 </epasd:AllocationUnits>
2346 <epasd:Connection>VS Network</epasd:Connection>
2347 <epasd:Description>Virtual NIC</epasd:Description>
2348 <epasd:ElementName>Ethernet Port</epasd:ElementName>
2349
2350 <epasd:InstanceID>3</epasd:InstanceID>
2351 <epasd:NetworkPortProfileID>aaaaaaaa-bbbb-cccc-dddd-
2352 eeeeeeeeeee</epasd:NetworkPortProfileID>
2353 <epasd:NetworkPortProfileIDType>3</epasd:NetworkPortProfileIDType>
2354 <epasd:Reservation>1</epasd:Reservation>
2355 <epasd:ResourceType>10</epasd:ResourceType>
2356 <epasd:VirtualQuantityUnits>1</epasd:VirtualQuantityUnits>
2357 </EthernetPortItem>
2358 <StorageItem>
2359 <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
2360 <sasd:Description>Virtual Disk</sasd:Description>
2361 <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
2362 <sasd:InstanceID>4</sasd:InstanceID>
2363 <sasd:Reservation>100</sasd:Reservation>
2364 <sasd:ResourceType>31</sasd:ResourceType>
2365 <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
2366 </StorageItem>
2367 </VirtualHardwareSection>
2368 <OperatingSystemSection ovf:id="58" ovf:required="false">
2369 <Info>Guest Operating System</Info>
2370 <Description>OS</Description>
2371 </OperatingSystemSection>
2372 </VirtualSystem>
2373 </Envelope>

```

## 2374 E.2 Example 2 (OVF descriptor for one virtual system and one network with a 2375 locally referenced network port profile)

2376 The example below shows an OVF descriptor that describes a virtual system and a network to which it  
2377 connects. The virtual system description in this example uses a network port profile that is described in a  
2378 local file that is contained in the same OVF package. The network described in the network section uses  
2379 the same network port profile description. The network port profile described in this example is used to  
2380 reserve 1 Gbps of bandwidth.

```

2381 <?xml version="1.0" encoding="UTF-8"?>
2382 <Envelope xsi:schemaLocation="http://schemas.dmtf.org/ovf/envelope/2
2383 file:///C:/dsp8023_2.0.0_wgv0.9.5.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2384 xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/2" xmlns="http://schemas.dmtf.org/ovf/envelope/2"
2385 xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
2386 xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
2387 xmlns:epasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2388 schema/2/CIM_EthernetPortAllocationSettingData"
2389 xmlns:sasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_StorageAllocationSettingData">
2390 <!-- References to all external files -->
2391 <References>
2392 <File ovf:id="file1" ovf:href="vmdisk1.vmdk" ovf:size="2000000000"/>
2393 <File ovf:id="networkportprofile1" ovf:href="NetworkPortProfile1.xml"/>
2394 </References>
2395 <!-- Describes meta-information for all virtual disks in the package -->
2396 <DiskSection>
2397 <Info>Describes the set of virtual disks</Info>
2398 <Disk ovf:diskId="vmdisk1" ovf:fileRef="file1" ovf:capacity="4294967296"
2399 ovf:format="http://www.examplecompany.com/interfaces/specifications/vmdk.html#sparse"/>
2400 </DiskSection>
2401 <!-- Describes all networks used in the package -->
2402 <NetworkSection>
2403 <Info>List of logical networks used in the package</Info>
2404 <Network ovf:name="VS Network">
2405 <Description>The network that VSs connect to</Description>
2406 <NetworkPortProfileURI>file:networkportprofile1</NetworkPortProfileURI>
2407 </Network>
2408 </NetworkSection>
2409 <VirtualSystem ovf:id="vm">
2410 <Info>Describes a virtual system</Info>
2411 <Name>Virtual Appliance One</Name>
2412 <ProductSection>
2413 <Info>Describes product information for the appliance</Info>
2414 <Product>The Great Appliance</Product>
2415 <Vendor>Some Great Corporation</Vendor>
2416 <Version>13.00</Version>
2417 <FullVersion>13.00-b5</FullVersion>
2418 <ProductUrl>http://www.somegreatcorporation.com/greatappliance</ProductUrl>
2419 <VendorUrl>http://www.somegreatcorporation.com/</VendorUrl>
2420 <Property ovf:key="adminemail" ovf:type="string">
2421 <Description>Email address of administrator</Description>
2422 </Property>
2423 <Property ovf:key="app_ip" ovf:type="string" ovf:defaultValue="192.168.0.10">
2424 <Description>The IP address of this appliance</Description>
2425 </Property>
2426 </ProductSection>
2427 <AnnotationSection ovf:required="false">
2428 <Info>A random annotation on this service. It can be ignored</Info>
2429 <Annotation>Contact customer support if you have any problems</Annotation>
2430 </AnnotationSection>
2431 <EulaSection>
2432 <Info>License information for the appliance</Info>
2433 <License>Insert your favorite license here</License>
2434 </EulaSection>
2435 <VirtualHardwareSection>
2436 <Info>Memory = 4 GB, CPU = 1 GHz, Disk = 100 GB, 1 Ethernet nic</Info>
2437 <Item>
2438 <rasd:AllocationUnits>Hertz*10^9</rasd:AllocationUnits>
2439 <rasd:Description>Virtual CPU</rasd:Description>

```



```

2440 <rasd:ElementName>1 GHz virtual CPU</rasd:ElementName>
2441 <rasd:InstanceID>1</rasd:InstanceID>
2442 <rasd:Reservation>1</rasd:Reservation>
2443 <rasd:ResourceType>3</rasd:ResourceType>
2444 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2445 </Item>
2446 <Item>
2447 <rasd:AllocationUnits>byte*2^30</rasd:AllocationUnits>
2448 <rasd:Description>Memory</rasd:Description>
2449 <rasd:ElementName>1 GByte of memory</rasd:ElementName>
2450 <rasd:InstanceID>2</rasd:InstanceID>
2451 <rasd:ResourceType>4</rasd:ResourceType>
2452 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2453 </Item>
2454 <EthernetPortItem>
2455 <epasd:Address>00-16-8B-DB-00-5E</epasd:Address>
2456 <epasd:Connection>VS Network</epasd:Connection>
2457 <epasd:Description>Virtual NIC</epasd:Description>
2458 <epasd:ElementName>Ethernet Port</epasd:ElementName>
2459
2460 <epasd:InstanceID>3</epasd:InstanceID>
2461 <epasd:NetworkPortProfileID>file:networkportprofile1</epasd:NetworkPortProfileID>
2462 <epasd:NetworkPortProfileIDType>2</epasd:NetworkPortProfileIDType>
2463 <epasd:ResourceType>10</epasd:ResourceType>
2464 <epasd:VirtualQuantityUnits>1</epasd:VirtualQuantityUnits>
2465 </EthernetPortItem>
2466 <StorageItem>
2467 <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
2468 <sasd:Description>Virtual Disk</sasd:Description>
2469 <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
2470 <sasd:InstanceID>4</sasd:InstanceID>
2471 <sasd:Reservation>100</sasd:Reservation>
2472 <sasd:ResourceType>31</sasd:ResourceType>
2473 <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
2474 </StorageItem>
2475 </VirtualHardwareSection>
2476 <OperatingSystemSection ovf:id="58" ovf:required="false">
2477 <Info>Guest Operating System</Info>
2478 <Description>OS</Description>
2479 </OperatingSystemSection>
2480 </VirtualSystem>
2481 </Envelope>

```

### 2482 **E.3 Example 3 (OVF descriptor for one virtual system and one network with a** 2483 **network port profile referenced by a URI)**

2484 The example below shows an OVF descriptor that describes a virtual system and a network to which it  
2485 connects. The virtual system description in this example uses a network port profile that is described by a  
2486 URI. The network described in the network section uses the same network port profile description. The  
2487 network port profile described in this example is used to reserve 1 Gbps of bandwidth.

```

2488 <?xml version="1.0" encoding="UTF-8"?>
2489 <Envelope xsi:schemaLocation="http://schemas.dmtf.org/ovf/envelope/2
2490 file:///C:/dsp8023_2.0.0_wgv0.9.5.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2491 xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/2" xmlns="http://schemas.dmtf.org/ovf/envelope/2"
2492 xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
2493 xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
2494 xmlns:epasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2495 schema/2/CIM_EthernetPortAllocationSettingData"
2496 xmlns:sasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_StorageAllocationSettingData">
2497 <!-- References to all external files -->
2498 <References>
2499 <File ovf:id="file1" ovf:href="vmdisk1.vmdk" ovf:size="2000000000"/>
2500 </References>
2501 <!-- Describes meta-information for all virtual disks in the package -->
2502 <DiskSection>
2503 <Info>Describes the set of virtual disks</Info>

```

```

2504 <Disk ovf:diskId="vmdisk1" ovf:fileRef="file1" ovf:capacity="4294967296"
2505 ovf:format="http://www.examplecompany.com/interfaces/specifications/vmdk.html#sparse"/>
2506 </DiskSection>
2507 <!-- Describes all networks used in the package -->
2508 <NetworkSection>
2509 <Info>List of logical networks used in the package</Info>
2510 <Network ovf:name="VS Network">
2511 <Description>The network that the VSs connect to</Description>
2512
2513 <NetworkPortProfileURI>http://www.dmtf.org/networkportprofiles/networkportprofile1.xml</Netwo
2514 rkPortProfileURI>
2515 </Network>
2516 </NetworkSection>
2517 <VirtualSystem ovf:id="vm">
2518 <Info>Describes a virtual system</Info>
2519 <Name>Virtual Appliance One</Name>
2520 <ProductSection>
2521 <Info>Describes product information for the appliance</Info>
2522 <Product>The Great Appliance</Product>
2523 <Vendor>Some Great Corporation</Vendor>
2524 <Version>13.00</Version>
2525 <FullVersion>13.00-b5</FullVersion>
2526 <ProductUrl>http://www.somegreatcorporation.com/greatappliance</ProductUrl>
2527 <VendorUrl>http://www.somegreatcorporation.com/</VendorUrl>
2528 <Property ovf:key="adminemail" ovf:type="string">
2529 <Description>Email address of administrator</Description>
2530 </Property>
2531 <Property ovf:key="app_ip" ovf:type="string" ovf:defaultValue="192.168.0.10">
2532 <Description>The IP address of this appliance</Description>
2533 </Property>
2534 </ProductSection>
2535 <AnnotationSection ovf:required="false">
2536 <Info>A random annotation on this service. It can be ignored</Info>
2537 <Annotation>Contact customer support if you have any problems</Annotation>
2538 </AnnotationSection>
2539 <EulaSection>
2540 <Info>License information for the appliance</Info>
2541 <License>Insert your favorite license here</License>
2542 </EulaSection>
2543 <VirtualHardwareSection>
2544 <Info>Memory = 4 GB, CPU = 1 GHz, Disk = 100 GB, 1 Ethernet nic</Info>
2545 <Item>
2546 <rasd:AllocationUnits>Hertz*10^9</rasd:AllocationUnits>
2547 <rasd:Description>Virtual CPU</rasd:Description>
2548 <rasd:ElementName>1 GHz virtual CPU</rasd:ElementName>
2549 <rasd:InstanceID>1</rasd:InstanceID>
2550 <rasd:Reservation>1</rasd:Reservation>
2551 <rasd:ResourceType>3</rasd:ResourceType>
2552 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2553 </Item>
2554 <Item>
2555 <rasd:AllocationUnits>byte*2^30</rasd:AllocationUnits>
2556 <rasd:Description>Memory</rasd:Description>
2557 <rasd:ElementName>1 GByte of memory</rasd:ElementName>
2558 <rasd:InstanceID>2</rasd:InstanceID>
2559 <rasd:ResourceType>4</rasd:ResourceType>
2560 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2561 </Item>
2562 <EthernetPortItem>
2563 <epasd:Address>00-16-8B-DB-00-5E</epasd:Address>
2564 <epasd:Connection>VS Network</epasd:Connection>
2565 <epasd:Description>Virtual NIC</epasd:Description>
2566 <epasd:ElementName>Ethernet Port</epasd:ElementName>
2567
2568 <epasd:InstanceID>3</epasd:InstanceID>
2569
2570 <epasd:NetworkPortProfileID>http://www.dmtf.org/networkportprofiles/networkportprofile1.xml</
2571 epasd:NetworkPortProfileID>
2572 <epasd:NetworkPortProfileIDType>2</epasd:NetworkPortProfileIDType>
2573 <epasd:ResourceType>10</epasd:ResourceType>

```

```

2574 <epas:VirtualQuantityUnits>1</epas:VirtualQuantityUnits>
2575 </EthernetPortItem>
2576 <StorageItem>
2577 <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
2578 <sasd:Description>Virtual Disk</sasd:Description>
2579 <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
2580 <sasd:InstanceID>4</sasd:InstanceID>
2581 <sasd:Reservation>100</sasd:Reservation>
2582 <sasd:ResourceType>31</sasd:ResourceType>
2583 <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
2584 </StorageItem>
2585 </VirtualHardwareSection>
2586 <OperatingSystemSection ovf:id="58" ovf:required="false">
2587 <Info>Guest Operating System</Info>
2588 <Description>OS</Description>
2589 </OperatingSystemSection>
2590 </VirtualSystem>
2591 </Envelope>

```

#### 2592 **E.4 Example 4 (OVF descriptor for two virtual systems and one network with** 2593 **two network port profiles referenced by URIs)**

2594 The example below shows an OVF descriptor that describes two virtual systems and a network to which  
2595 they connect. Each virtual system description in this example uses a network port profile that is described  
2596 by a URI. The network described in the network section uses the same two network port profiles. The two  
2597 network port profiles described in this example are used to reserve 1 Gbps of bandwidth and describe  
2598 general network traffic respectively. Annex E.5 and E.6 are examples of these network port profiles.

```

2599 <?xml version="1.0" encoding="UTF-8"?>
2600 <Envelope xsi:schemaLocation="http://schemas.dmtf.org/ovf/envelope/2
2601 file:///C:/dsp8023 2.0.0 wgv0.9.5.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2602 xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/2" xmlns="http://schemas.dmtf.org/ovf/envelope/2"
2603 xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
2604 xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
2605 xmlns:epasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2606 schema/2/CIM_EthernetPortAllocationSettingData"
2607 xmlns:sasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_StorageAllocationSettingData">
2608 <!-- References to all external files -->
2609 <References>
2610 <File ovf:id="file1" ovf:href="vmdisk1.vmdk" ovf:size="2000000000"/>
2611 </References>
2612 <!-- Describes meta-information for all virtual disks in the package -->
2613 <DiskSection>
2614 <Info>Describes the set of virtual disks</Info>
2615 <Disk ovf:diskId="vmdisk1" ovf:fileRef="file1" ovf:capacity="4294967296"
2616 ovf:format="http://www.examplecompany.com/interfaces/specifications/vmdk.html#sparse"/>
2617 </DiskSection>
2618 <!-- Describes all networks used in the package -->
2619 <NetworkSection>
2620 <Info>List of logical networks used in the package</Info>
2621 <Network ovf:name="VS Network">
2622 <Description>The network that the VSs connect to</Description>
2623 <!-- Network port profile for storage traffic -->
2624
2625 <NetworkPortProfileURI>http://www.dmtf.org/networkportprofiles/networkportprofile1.xml</Netwo
2626 rkPortProfileURI>
2627 <!-- Network port profile for networking traffic -->
2628
2629 <NetworkPortProfileURI>http://www.dmtf.org/networkportprofiles/networkportprofile2.xml</Netwo
2630 rkPortProfileURI>
2631 </Network>
2632 </NetworkSection>
2633 <VirtualSystemCollection ovf:id="vscl">
2634 <Info>Collection of 2 VSs</Info>
2635 <VirtualSystem ovf:id="storage server">
2636 <Info>Describes a virtual system</Info>
2637 <Name>Virtual Appliance One</Name>
2638 <ProductSection>

```

```

2639 <Info>Describes product information for the appliance</Info>
2640 <Product>The Great Appliance</Product>
2641 <Vendor>Some Great Corporation</Vendor>
2642 <Version>13.00</Version>
2643 <FullVersion>13.00-b5</FullVersion>
2644 <ProductUrl>http://www.somegreatcorporation.com/greatappliance</ProductUrl>
2645 <VendorUrl>http://www.somegreatcorporation.com/</VendorUrl>
2646 <Property ovf:key="adminemail" ovf:type="string">
2647 <Description>Email address of administrator</Description>
2648 </Property>
2649 <Property ovf:key="app_ip" ovf:type="string" ovf:defaultValue="192.168.0.10">
2650 <Description>The IP address of this appliance</Description>
2651 </Property>
2652 </ProductSection>
2653 <AnnotationSection ovf:required="false">
2654 <Info>A random annotation on this service. It can be ignored</Info>
2655 <Annotation>Contact customer support if you have any problems</Annotation>
2656 </AnnotationSection>
2657 <EulaSection>
2658 <Info>License information for the appliance</Info>
2659 <License>Insert your favorite license here</License>
2660 </EulaSection>
2661 <VirtualHardwareSection>
2662 <Info>Memory = 4 GB, CPU = 1 GHz, Disk = 100 GB, 1 Ethernet nic</Info>
2663 <Item>
2664 <rasd:AllocationUnits>Hertz*10^9</rasd:AllocationUnits>
2665 <rasd:Description>Virtual CPU</rasd:Description>
2666 <rasd:ElementName>1 GHz virtual CPU</rasd:ElementName>
2667 <rasd:InstanceID>1</rasd:InstanceID>
2668 <rasd:Reservation>1</rasd:Reservation>
2669 <rasd:ResourceType>3</rasd:ResourceType>
2670 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2671 </Item>
2672 <Item>
2673 <rasd:AllocationUnits>byte*2^30</rasd:AllocationUnits>
2674 <rasd:Description>Memory</rasd:Description>
2675 <rasd:ElementName>1 GByte of memory</rasd:ElementName>
2676 <rasd:InstanceID>2</rasd:InstanceID>
2677 <rasd:ResourceType>4</rasd:ResourceType>
2678 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2679 </Item>
2680 <EthernetPortItem>
2681 <epasd:Address>00-16-8B-DB-00-5E</epasd:Address>
2682 <epasd:Connection>VS Network</epasd:Connection>
2683 <epasd:Description>Virtual NIC</epasd:Description>
2684
2685 <epasd:ElementName>Ethernet Port</epasd:ElementName>
2686
2687 <epasd:InstanceID>3</epasd:InstanceID>
2688
2689 <epasd:NetworkPortProfileID>http://www.dmtf.org/networkportprofiles/networkportprofile1.xml</
2690 epasd:NetworkPortProfileID>
2691 <epasd:NetworkPortProfileIDType>2</epasd:NetworkPortProfileIDType>
2692 <epasd:ResourceType>10</epasd:ResourceType>
2693 <epasd:VirtualQuantityUnits>1</epasd:VirtualQuantityUnits>
2694 </EthernetPortItem>
2695 <StorageItem>
2696 <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
2697 <sasd:Description>Virtual Disk</sasd:Description>
2698 <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
2699 <sasd:InstanceID>4</sasd:InstanceID>
2700 <sasd:Reservation>100</sasd:Reservation>
2701 <sasd:ResourceType>31</sasd:ResourceType>
2702 <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
2703 </StorageItem>
2704 </VirtualHardwareSection>
2705 <OperatingSystemSection ovf:id="58" ovf:required="false">
2706 <Info>Guest Operating System</Info>
2707 <Description>OS</Description>
2708 </OperatingSystemSection>

```

```

2709 </VirtualSystem>
2710 <VirtualSystem ovf:id="web-server">
2711 <Info>Describes a virtual system</Info>
2712 <Name>Virtual Appliance Two</Name>
2713 <ProductSection>
2714 <Info>Describes product information for the appliance</Info>
2715 <Product>The Great Appliance</Product>
2716 <Vendor>Some Great Corporation</Vendor>
2717 <Version>13.00</Version>
2718 <FullVersion>13.00-b5</FullVersion>
2719 <ProductUrl>http://www.somegreatcorporation.com/greatappliance</ProductUrl>
2720 <VendorUrl>http://www.somegreatcorporation.com/</VendorUrl>
2721 <Property ovf:key="adminemail" ovf:type="string">
2722 <Description>Email address of administrator</Description>
2723 </Property>
2724 <Property ovf:key="app ip" ovf:type="string" ovf:defaultValue="192.168.0.10">
2725 <Description>The IP address of this appliance</Description>
2726 </Property>
2727 </ProductSection>
2728 <AnnotationSection ovf:required="false">
2729 <Info>A random annotation on this service. It can be ignored</Info>
2730 <Annotation>Contact customer support if you have any problems</Annotation>
2731 </AnnotationSection>
2732 <EulaSection>
2733 <Info>License information for the appliance</Info>
2734 <License>Insert your favorite license here</License>
2735 </EulaSection>
2736 <VirtualHardwareSection>
2737 <Info>Memory = 4 GB, CPU = 1 GHz, Disk = 100 GB, 1 Ethernet nic</Info>
2738 <Item>
2739 <rasd:AllocationUnits>Hertz*10^9</rasd:AllocationUnits>
2740 <rasd:Description>Virtual CPU</rasd:Description>
2741 <rasd:ElementName>1 GHz virtual CPU</rasd:ElementName>
2742 <rasd:InstanceID>1</rasd:InstanceID>
2743 <rasd:Reservation>1</rasd:Reservation>
2744 <rasd:ResourceType>3</rasd:ResourceType>
2745 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2746 </Item>
2747 <Item>
2748 <rasd:AllocationUnits>byte*2^30</rasd:AllocationUnits>
2749 <rasd:Description>Memory</rasd:Description>
2750 <rasd:ElementName>1 GByte of memory</rasd:ElementName>
2751 <rasd:InstanceID>2</rasd:InstanceID>
2752 <rasd:ResourceType>4</rasd:ResourceType>
2753 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2754 </Item>
2755 <EthernetPortItem>
2756 <epasd:Address>00-16-8B-DB-00-5F</epasd:Address>
2757 <epasd:Connection>VS Network</epasd:Connection>
2758 <epasd:Description>Virtual NIC</epasd:Description>
2759
2760 <epasd:ElementName>Ethernet Port</epasd:ElementName>
2761 <!-- Virtual NIC for networking traffic -->
2762 <epasd:InstanceID>3</epasd:InstanceID>
2763
2764 <epasd:NetworkPortProfileID>http://www.dmtf.org/networkportprofiles/networkportprofile2.xml</
2765 epasd:NetworkPortProfileID>
2766 <epasd:NetworkPortProfileIDType>2</epasd:NetworkPortProfileIDType>
2767 <epasd:ResourceType>10</epasd:ResourceType>
2768 <epasd:VirtualQuantityUnits>1</epasd:VirtualQuantityUnits>
2769 </EthernetPortItem>
2770 <StorageItem>
2771 <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
2772 <sasd:Description>Virtual Disk</sasd:Description>
2773 <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
2774 <sasd:InstanceID>4</sasd:InstanceID>
2775 <sasd:Reservation>100</sasd:Reservation>
2776 <sasd:ResourceType>31</sasd:ResourceType>
2777 <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
2778 </StorageItem>

```

```

2779 </VirtualHardwareSection>
2780 <OperatingSystemSection ovf:id="58" ovf:required="false">
2781 <Info>Guest Operating System</Info>
2782 <Description>OS</Description>
2783 </OperatingSystemSection>
2784 </VirtualSystem>
2785 </VirtualSystemCollection>
2786 </Envelope>

```

## 2787 E.5 Example 5 (networkportprofile1.xml)

2788  
2789 Network port profile example for bandwidth reservation.

```

2790 <?xml version="1.0" encoding="UTF-8"?>
2791 <NetworkPortProfile xsi:schemaLocation="http://schemas.dmtf.org/ovf/networkportprofile/1
2792 http://schemas.dmtf.org/ovf/networkportprofile/1/dsp8049.xsd"
2793 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2794 xmlns="http://schemas.dmtf.org/ovf/networkportprofile/1"
2795 xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
2796 xmlns:epasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2797 schema/2/CIM_EthernetPortAllocationSettingData">
2798 <Item>
2799 <epasd:AllocationUnits>bit / second * 10^9</epasd:AllocationUnits>
2800 <epasd:ElementName>Network Port Profile 1</epasd:ElementName>
2801 <epasd:InstanceID>1</epasd:InstanceID>
2802 <epasd:NetworkPortProfileID>aaaaaaaa-bbbb-cccc-dddd-
2803 eeeeeeeeeeee</epasd:NetworkPortProfileID>
2804 <epasd:NetworkPortProfileIDType>3</epasd:NetworkPortProfileIDType>
2805 <epasd:Reservation>1</epasd:Reservation>
2806 </Item>
2807 </NetworkPortProfile>

```

## 2808 E.6 Example 6 (networkportprofile2.xml)

2809  
2810 Network port profile example showing priority setting.

```

2811 <?xml version="1.0" encoding="UTF-8"?>
2812 <NetworkPortProfile xsi:schemaLocation="http://schemas.dmtf.org/ovf/networkportprofile/1
2813 http://schemas.dmtf.org/ovf/networkportprofile/1/dsp8049.xsd"
2814 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2815 xmlns="http://schemas.dmtf.org/ovf/networkportprofile/1"
2816 xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
2817 xmlns:epasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2818 schema/2/CIM_EthernetPortAllocationSettingData">
2819 <Item>
2820 <epasd:AllowedPriorities>0</epasd:AllowedPriorities>
2821 <epasd:AllowedPriorities>1</epasd:AllowedPriorities>
2822 <epasd:DefaultPriority>0</epasd:DefaultPriority>
2823 <epasd:ElementName>Network Port Profile 2</epasd:ElementName>
2824 <epasd:InstanceID>2</epasd:InstanceID>
2825 <epasd:NetworkPortProfileID>aaaaaaaa-bbbb-cccc-dddd-
2826 ffffffff</epasd:NetworkPortProfileID>
2827 <epasd:NetworkPortProfileIDType>3</epasd:NetworkPortProfileIDType>
2828 </Item>
2829 </NetworkPortProfile>

```

## ANNEX F (informative)

2830  
2831  
2832  
2833

### Deployment considerations

2834 This standard defines an OVF package and the main clauses in this standard deal with this subject  
2835 matter. However, there are deployment considerations necessary to meet the expectations of the OVF  
2836 package author. These are listed below.

#### 2837 **F.1 OVF package structure deployment considerations**

2838 A deployment function shall verify the ovf package signature and should validate the certificate.

#### 2839 **F.2 Virtual hardware deployment considerations**

2840 If there are multiple virtual hardware sections, the deployment function should select the most appropriate  
2841 one for the target virtualization platform.

2842 If no backing is specified for a device that requires a backing, the deployment function shall make an  
2843 appropriate choice, for example, by prompting the user. More than one backing for a device shall not be  
2844 specified.

2845 The deployment function should select the normal value for a resource allocation but may adjust it within  
2846 the specified range. The virtualization management may further alter the resource allocation within the  
2847 specified range for performance tuning.

#### 2848 **F.3 Core metadata sections deployment considerations**

2849 The sharing of disk blocks at runtime is optional and virtualization platform specific and shall not be visible  
2850 to the guest software.

2851 A virtualization platform may share storage extents to minimize the amount of space required to support  
2852 the virtual systems. If storage extents are shared by the virtualization platform, this sharing is not visible to  
2853 the guest software.

2854 If present, the AnnotationSection element may be displayed during deployment of the OVF package.

2855 If present, the EULASection(s) shall be displayed and accepted during deployment of an OVF package. If  
2856 automated deployment is used, the deployment function shall have a methodology to provide implicit  
2857 acceptance.

2858 If virtual disks or other files are included by reference, the deployment function shall acquire those files  
2859 prior to the virtual system being launched.

2860 If the specified boot source is a device type, the deployment function should try all the devices of that  
2861 device type specified.

**ANNEX G  
(informative)****Change log**2862  
2863  
2864  
2865

| Version | Date       | Description                                                                                                                                                                                                                                                   |
|---------|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1.0.0   | 2009-02-22 | DMTF Standard release                                                                                                                                                                                                                                         |
| 1.1.0   | 2010-01-12 | DMTF Standard release                                                                                                                                                                                                                                         |
| 2.0.0   | 2012-10-29 | DMTF Standard release                                                                                                                                                                                                                                         |
| 2.1.0   | 2013-12-12 | DMTF Standard release (fixed "." For ovf:key values)<br>Line 1829, 2331, 2442, 2550, 2668, 2743 admin.email -> adminemail<br>Line 1904 app.adminEmail -> app_adminEmail<br>Line 1909 app.log -> app_log<br>Line 2334, 2445, 2553, 2671, 2746 app.ip -> app_ip |

2866