



1
2
3
4

Document Number: DSP0243

Date: 2013-07-02

Version: 2.1.0c

5 **Open Virtualization Format Specification**

Information for Work-in-Progress version:

IMPORTANT: This document is not a standard. It does not necessarily reflect the views of the DMTF or all of its members. Because this document is a Work in Progress, it may still change, perhaps profoundly. This document is available for public review and comment until the stated expiration date.

It expires on: 2013-12-31

**Provide any comments through the DMTF Feedback Portal:
<http://www.dmtf.org/standards/feedback>**

6 **Document Type: Specification**
7 **Document Status: Work in Progress**
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25 withdrawn or modified after publication, and shall be indemnified and held harmless by any party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>.

CONTENTS

32	Foreword	6
33	Introduction.....	7
34	1 Scope	8
35	2 Normative References.....	8
36	3 Terms and Definitions	9
37	4 Symbols and Abbreviated Terms	12
38	5 OVF Package	12
39	5.1 OVF Package Structure	12
40	5.2 Virtual Disk Formats.....	13
41	5.3 OVF PackageOptions	13
42	5.4 Distribution as a Set of Files	14
43	6 OVF Descriptor.....	14
44	7 Envelope Element	15
45	7.1 File References	15
46	7.2 Content Element	16
47	7.3 Extensibility	17
48	7.4 Conformance	17
49	8 Virtual Hardware Description.....	18
50	8.1 VirtualHardwareSection	18
51	8.2 Extensibility	19
52	8.3 Virtual Hardware Elements	20
53	8.4 Ranges on Elements.....	21
54	9 Core Metadata Sections in version 2	23
55	9.1 DiskSection	24
56	9.2 NetworkSection.....	25
57	9.3 ResourceAllocationSection	25
58	9.4 AnnotationSection.....	26
59	9.5 ProductSection.....	26
60	9.5.1 Property Elements	27
61	9.6 EulaSection.....	29
62	9.7 StartupSection	29
63	9.8 DeploymentOptionSection	30
64	9.9 OperatingSystemSection	31
65	9.10 InstallSection.....	31
66	9.11 EnvironmentFilesSection	32
67	9.12 BootDeviceSection.....	32
68	9.13 SharedDiskSection	33
69	9.14 ScaleOutSection	33
70	9.15 PlacementGroupSection and PlacementSection.....	34
71	9.16 Encryption Section	36
72	10 Internationalization	37
73	10.1 Internal Resource Bundles	37
74	10.2 External Resource Bundles	37
75	10.3 Message Content in External File.....	37
76	11 OVF Environment and OVF Environment File	38
77	11.1 Transport Media.....	39
78	11.2 Transport Media Type.....	39
79	ANNEX A (informative) Symbols and Conventions	41
80	ANNEX B (normative) OVF XSD	42
81	ANNEX C (informative) OVF Mime Type Registration Template	43

82	ANNEX D (informative) OVF Examples	45
83	D.1 Examples of OVF Package Structure	45
84	D.2 Examples of Distribution of Files	45
85	D.3 Example of Envelope Element.....	46
86	D.4 Example of File References.....	47
87	D.5 Example of Content Element	47
88	D.6 Examples of Extensibility	47
89	D.7 Examples of VirtualHardwareSection	48
90	D.8 Examples of Virtual Hardware Elements	49
91	D.9 Example of Ranges on Elements	49
92	D.10 Example of DiskSection	50
93	D.11 Example of NetworkSection.....	50
94	D.12 Example of ResourceAllocationSection	51
95	D.13 Example of Annotation.....	51
96	D.14 Example of Product Section	51
97	D.15 Example of EULA Section	52
98	D.16 Example of StartupSection	52
99	D.17 Example of DeploymentOptionSection	52
100	D.18 Example of OperatingSystemSection	53
101	D.19 Example of InstallSection	53
102	D.20 Example of EnvironmentFilesSection	54
103	D.21 Example of BootDeviceSection	54
104	D.22 Example of SharedDiskSection	55
105	D.23 Example of ScaleOutSection	55
106	D.24 Example of PlacementGroupSection.....	56
107	D.25 Example of EncryptionSection.....	57
108	D.26 Example of Internationalization.....	58
109	D.27 Example of Message Content in an External File.....	59
110	D.28 Example of Environment Document	60
111	ANNEX E (informative) Network Port Profile Examples	61
112	E.1 Example 1 (OVF Descriptor for One Virtual System and One Network with an Inlined Network Port Profile).....	61
113	E.2 Example 2 (OVF Descriptor for One Virtual System and One Network with a Locally Referenced Network Port Profile)	63
114	E.3 Example 3 (OVF Descriptor for One Virtual System and One Network with a Network Port Profile referenced by a URI)	64
115	E.4 Example 4 (OVF Descriptor for Two Virtual Systems and One Network with Two Network Port Profiles referenced by URIs)	66
116	E.5 Example 5 (networkportprofile1.xml)	69
117	E.6 Example 6 (networkportprofile2.xml)	69
118	ANNEX F (informative) Deployment Considerations	70
119	F.1 OVF Package Structure Deployment Considerations	70
120	F.2 Virtual Hardware Deployment Considerations.....	70
121	F.3 Core Metadata Sections Deployment Considerations.....	70
122	ANNEX G (informative) Bibliography	71
123	ANNEX H (informative) Change Log	72
124		
125		
126		
127		
128		

129 **Tables**

130 Table 1 – XML Namespace Prefixes 15

131 Table 2 – Actions for Child Elements with `ovf:required` Attribute..... 19

132 Table 3 – HostResource Element 20

133 Table 4 – Elements for Virtual Devices and Controllers 21

134 Table 5 – Core Metadata Sections 23

135 Table 6 – Property Types 28

136 Table 7 – Property Qualifiers 28

137 Table 8 – Availability Attributes 35

138 Table 9 – Core Sections..... 35

139 Table 10 – Allowed Combinations of Scoped Affinity and Availability 36

140 Table 11 – Core Sections..... 39

141

142

Foreword

143 The *Open Virtualization Format Specification* (DSP0243) was prepared by the OVFWork Group of the
144 DMTF.

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

147

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

186

Introduction

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

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

191 • **Optimized for distribution**

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

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

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

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

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

202 • **Portable VS packaging**

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

207 • **Vendor and platform independent**

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

210 • **Extensible**

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

214 • **Localizable**

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

218 • **Open standard**

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

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

223

Open Virtualization Format Specification

224 1 Scope

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

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

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

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

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

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

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

241 2 Normative References

242 The following referenced documents are indispensable for the application of this document. For dated
243 references, only the edition cited applies. For undated references, the latest edition of the referenced
244 document (including any amendments) applies.

245 [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
246 Specifications, Issue 7

247 http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50516

248 DMTF DSP0004, *Common Information Model (CIM) Infrastructure Specification 2.7*,

249 http://www.dmtf.org/standards/published_documents/DSP0004_2.7.pdf

250 DMTF DSP0230, *WS-CIM Mapping Specification 1.0*,

251 http://www.dmtf.org/standards/published_documents/DSP0230_1.0.pdf

252 DMTF DSP1041, *Resource Allocation Profile (RAP) 1.1*,

253 http://www.dmtf.org/standards/published_documents/DSP1041_1.1.pdf

254 DMTF DSP1043, *Allocation Capabilities Profile (ACP) 1.0*,

255 http://www.dmtf.org/standards/published_documents/DSP1043_1.0.pdf

256 DMTF DSP1047, *Storage Resource Virtualization Profile 1.0*,

257 http://www.dmtf.org/standards/published_documents/DSP1047_1.0.pdf

258 DMTF DSP1057, *Virtual System Profile 1.0*,

259 http://www.dmtf.org/standards/published_documents/DSP1057_1.0.pdf

- 260 DMTF DSP8023, *Open Virtualization Format (OVF) 2 XML Schema*,
261 <http://schemas.dmtf.org/ovf/envelope/2/dsp8023.xsd>
- 262 DMTF DSP8049, *Network Port Profile XML Schema*,
263 <http://schemas.dmtf.org/ovf/networkportprofile/1/dsp8049.xsd>
- 264 IETF RFC1738, T. Berners-Lee, *Uniform Resource Locators (URL)*, December 1994,
265 <http://tools.ietf.org/html/rfc1738>
- 266 IETF RFC1952, P. Deutsch, *GZIP file format specification version 4.3*, May 1996,
267 <http://tools.ietf.org/html/rfc1952>
- 268 IETF Standard 68, *Augmented BNF for Syntax Specifications: ABNF*,
269 <http://tools.ietf.org/html/rfc5234>
- 270 IETF RFC2616, R. Fielding et al, *Hypertext Transfer Protocol – HTTP/1.1*, June 1999,
271 <http://tools.ietf.org/html/rfc2616>
- 272 IETF Standard 66, *Uniform Resource Identifiers (URI): Generic Syntax*,
273 <http://tools.ietf.org/html/rfc3986>
- 274 ISO 9660, 1988 Information processing-Volume and file structure of CD-ROM for information interchange,
275 http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=17505
- 276 ISO, ISO/IEC Directives, Part 2, *Rules for the structure and drafting of International Standards*,
277 <http://isotc.iso.org/livelink/livelink.exe?func=ll&objId=4230456&objAction=browse&sort=subtype>
- 278 W3C, *XML Schema Part 1: Structures Second Edition*, 28 October 2004. W3C Recommendation. URL:
279 <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/>
- 280 W3C, *XML Schema Part 2: Datatypes Second Edition*, 28 October 2004. W3C Recommendation. URL:
281 <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/>
- 282 XML Encryption Syntax and Processing Version 1.1, 13 March 2012, W3C Candidate Recommendation
283 <http://www.w3.org/TR/2012/CR-xmlenc-core1-20120313/>
- 284 FIPS 180-2: Secure Hash Standard (SHS)
285 http://www.nist.gov/manuscript-publication-search.cfm?pub_id=902003#

286 **3 Terms and Definitions**

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

288 **3.1**

289 **authoring function**

290 the creation of the OVF package

291 **3.2**

292 **can**

293 used for statements of possibility and capability, whether material, physical, or causal

294 **3.3**

295 **cannot**

296 used for statements of possibility and capability, whether material, physical, or causal

- 297 **3.4**
298 **chassis**
299 a placement policy as defined in the class CIM_Chassis
- 300 **3.5**
301 **conditional**
302 indicates requirements to be followed strictly to conform to the document when the specified conditions
303 are met
- 304 **3.6**
305 **deployment function**
306 a function the result of which is a prepared virtual system
- 307 **3.7**
308 **geographic**
309 a placement policy referring to a geographic location (e.g., a country, a state, a province, a latlong)
- 310 **3.8**
311 **guest software**
312 the software that runs inside a virtual system
- 313 **3.9**
314 **mandatory**
315 indicates requirements to be followed strictly to conform to the document and from which no deviation is
316 permitted
- 317 **3.10**
318 **may**
319 indicates a course of action permissible within the limits of the document
- 320 **3.11**
321 **need not**
322 indicates a course of action permissible within the limits of the document
- 323 **3.12**
324 **optional**
325 indicates a course of action permissible within the limits of the document
- 326 **3.13**
327 **rack**
328 a placement policy as defined as defined in the class CIM_Rack
- 329 **3.14**
330 **shall**
331 indicates requirements to be followed strictly to conform to the document and from which no deviation is
332 permitted
- 333 **3.15**
334 **shall not**
335 indicates requirements to be followed strictly to conform to the document and from which no deviation is
336 permitted

- 337 **3.16**
338 **should**
339 indicates that among several possibilities, one is recommended as particularly suitable, without
340 mentioning or excluding others, or that a certain course of action is preferred but not necessarily required
- 341 **3.17**
342 **should not**
343 indicates that a certain possibility or course of action is deprecated but not prohibited
- 344 **3.18**
345 **site**
346 a placement policy as defined in Access, Terminals, Transmission and Multiplexing (ATTM); Broadband
347 Deployment - Energy Efficiency and Key Performance Indicators; Part 2: Network sites; Sub-part 1:
348 Operator sites, Technical Report, ETSI TR 105 174-2-1 V1.1.1 (2009-10)
- 349 **3.19**
350 **OVF package**
351 a single compressed file or a set of files that contains the OVF descriptor file and may contain associated
352 virtual disks, operational metadata, and other files.
- 353 **3.20**
354 **OVF descriptor**
355 an XML file that validates to DSP8023 and provides the information needed to deploy the OVF package
- 356 **3.21**
357 **virtualization platform**
358 the hypervisor that the virtual systems run on
- 359 **3.22**
360 **virtual appliance**
361 a service delivered as a software stack that utilizes one or more virtual systems
- 362 **3.23**
363 **virtual hardware**
364 the processor, memory and I/O resources provided by a virtualization platform that supports a virtual
365 system
- 366 **3.24 virtual system**
367 **virtual system**
368 as defined in the Virtual System Profile plus the guest software if any. virtual system
- 369 **3.25**
370 **virtual system collection**
371 a collection of virtual systems
- 372 **3.26**
373 **virtualization management**
374 the software that performs resource allocation and management of virtual systems

375 4 Symbols and Abbreviated Terms

376 The following symbols and abbreviations are used in this document.

377 4.1

378 **CIM**

379 Common Information Model

380 4.2

381 **IP**

382 Internet Protocol

383 4.3

384 **OVF**

385 Open Virtualization Format

386 4.4

387 **VS**

388 Virtual system

389 4.5

390 **VSC**

391 Virtual system collection

392

393 5 OVF Package

394 5.1 OVF Package Structure

395 An OVF package shall consist of the following files:

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

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

402 An OVF package can be stored as either a single compressed file (`.ova`) or a set of files, as described in
403 5.3 and 5.4. Both modes shall be supported.

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

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

412 The format of the manifest file is as follows:

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

421 See D.1 for an example.

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

425 See ANNEX F for deployment considerations.

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

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

441 See D.1 for an example.

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

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

449 5.2 Virtual Disk Formats

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

456 5.3 OVF PackageOptions

457 An OVF package may be stored as a compressed OVF package or as a set of files in a directory
458 structure.

459 A compressed OVF package is stored as single file. The file extension is `.ova` (open virtual appliance or
460 application) See D.2 for an example.

461 All file references in the OVF descriptor shall be relative-path references and shall point to files included
462 in the compressed OVF package. Relative directories inside are allowed, but relative-path references
463 shall not contain “..” dot-segments.

464 Entries in a compressed OVF package shall exist only once.

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

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

470 or

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

477 or

478 1) OVF descriptor
479 2) The intermediate files shall be in the same order as listed in the References section (see
480 7.1). Note that any external string resource bundle files for internationalization shall be
481 first in the References section (see clause 10).
482 3) OVF manifest
483 4) OVF certificate

484 The ordering restriction ensures that it is possible to extract the OVF descriptor from a compressed OVF
485 package without scanning the entire archive. The ordering restriction ensures that a compressed OVF
486 package can be generated on-the-fly.

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

489 5.4 Distribution as a Set of Files

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

491 6 OVF Descriptor

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

494 The DMTF DSP8023 schema definition file for the OVF descriptor contains the elements and attributes.
495 The OVF descriptor shall validate with the DMTF DSP8023.

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

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

501

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

502 7 Envelope Element

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

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

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

515 See D.3 for an example.

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

519 7.1 File References

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

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

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

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

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

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

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

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

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

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

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

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

564 7.2 Content Element

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

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

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

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

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

582 7.3 Extensibility

583 Custom meta-data may be added to OVF descriptors in several ways:

584 • New section elements may be defined as part of the `Section` substitution group, and used
585 where the OVF schemas allow sections to be present. All subtypes of the `Section` element shall
586 contain an `Info` element that contains a human readable description of the meaning of this
587 entity. The values of `Info` elements can be used, for example, to give meaningful warnings to
588 users when a section is being skipped, even if the parser does not know anything about the
589 section. Clause 10 defines how to localize the `Info` element.

590 • The OVF schemas use an open content model, where all existing types may be extended at the
591 end with additional elements. Extension points are declared in the OVF schemas with `xs:any`
592 declarations with `namespace="##other"`.

593 • The OVF schemas allow additional attributes on existing types.

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

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

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

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

604 See D.6 for an example.

605 7.4 Conformance

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

608 • the OVF descriptor uses only sections and elements and attributes that are defined in this
609 specification.
610 Conformance Level: 1.

611 • the OVF descriptor uses custom sections or elements or attributes that are not defined in this
612 specification and all such extensions are optional as defined in 7.3.
613 Conformance Level: 2.

614 • the OVF descriptor uses custom sections or elements that are not defined in this specification
615 and at least one such extension is required as defined in 7.3. The definition of all required
616 extensions shall be publicly available in an open and unencumbered XML Schema. The complete
617 specification may be inclusive in the XML schema or available as a separate document.
618 Conformance Level: 3.

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

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

622 **8 Virtual Hardware Description**

623 **8.1 VirtualHardwareSection**

624 The `VirtualHardwareSection` element can be used to describe the virtual hardware used by the virtual
625 system.

626 This standard allows incomplete virtual hardware descriptions.

627 The virtualization platform may create additional virtual hardware devices.

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

629 This virtual hardware description is based on the CIM classes `CIM_VirtualSystemSettingData`,
630 `CIM_ResourceAllocationSettingData`, `CIM_EthernetPortAllocationSettingData`, and
631 `CIM_StorageAllocationSettingData`. The XML representation of the CIM model is based on the
632 WS-CIM mapping (DSP0230).
633 Note: This means that the XML elements that belong to the class complex type should be ordered by
634 Unicode code point (binary) order of their CIM property name identifiers. See D.7 for an example.

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

636 A `VirtualSystem` element shall have a `VirtualHardwareSection` direct child element. The
637 `VirtualHardwareSection` shall not be a direct child element of a `VirtualSystemCollection`
638 element or of an `Envelope` element.

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

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

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

648 A `VirtualHardwareSection` element shall have ofhe following direct child elements:

649 • zero or one `System` elements

650 • zero or more `Item` elements

- 651 • zero or more `EthernetPortItem` elements
- 652 • zero or more `StorageItem` elements.

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

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

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

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

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

674 8.2 Extensibility

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

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

681 **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

682 8.3 Virtual Hardware Elements

683 The element type of the `Item` element in a `VirtualHardwareSection` element is
 684 `CIM_ResourceAllocationSettingData_Type` as defined in `CIM_ResourceAllocationSettingData` .
 685 See ANNEX B.

686 The child elements of `Item` represent the values of one or more properties exposed by the
 687 `CIM_ResourceAllocationSettingData` class. They have the semantics of defined settings as
 688 defined in DSP1041, any profiles derived from DSP1041 for specific resource types, and this standard.
 689 See D.8 for an example.

690 The element type of the `EthernetPortItem` element in a `VirtualHardwareSection` element is
 691 `CIM_EthernetPortAllocationSettingData_Type` as defined in
 692 `CIM_EthernetPortAllocationSettingData`. See ANNEX B.

693 The child elements represent the values of one or more properties exposed by the
 694 `CIM_EthernetPortAllocationSettingData` class. They have the semantics of defined settings as
 695 defined in DSP1050, any profiles derived from DSP1050 for specific Ethernet port resource types, and
 696 this standard. See D.8 for an example.

697 The element type of the `StorageItem` element in a `VirtualHardwareSection` element is
 698 `CIM_StorageAllocationSettingData_Type` as defined in `CIM_StorageAllocationSettingData`. See
 699 ANNEX B

700 The child elements represent the values of one or more properties exposed by the
 701 `CIM_StorageAllocationSettingData` class. They have the semantics of defined settings as defined
 702 in DSP1047, any profiles derived from DSP1047 for specific storage resource types, and this standard.
 703 See D.8 for an example.

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

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

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

714 All Ethernet adapters in the OVF package that connect to the same network shall have a `Connection`
 715 element that contains the same logical network name. If a `Connection` element is used to represent a
 716 network then the corresponding network shall be represented as a child element of the `Network Section`
 717 element with a `name` attribute that matches the value of the `Connection` element.

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

721 **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.

ovf:/disk/<id>	A reference to a virtual disk, as specified in the DiskSection or SharedDiskSection. <id> shall be the value of the ovf:diskId attribute of the Disk element being referenced.
----------------	--

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

724 Table 4 gives a brief overview on how elements from rasd, epasd, and sasd namespaces are used to
 725 describe virtual devices and controllers.

726 **Table 4 – Elements for Virtual Devices and Controllers**

Element	Usage
Description	A human-readable description of the meaning of the information. For example, “Specifies the memory size of the virtual system”.
ElementName	A human-readable description of the content..
InstanceID	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, this element specifies whether the device should be connected at power on.
Parent	The InstanceID of the parent controller (if any).
Connection	Used with Ethernet adaptors to specify the network connection name for the virtual system.
Address	Device specific.
AddressOnParent	For a device, this 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.

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

730 **8.4 Ranges on Elements**

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

734 See ANNEX F for virtual hardware deployment considerations.

735 For the `Item`, `EthernetPortItem`, and `StorageItem` elements in the
736 `VirtualHardwareSection` and the `ResourceAllocationSection` elements, the following
737 additional semantics are defined:

- 738 • Each `Item`, `EthernetPortItem`, or `StorageItem` element has an optional `ovf:bound`
739 attribute. This value may be specified as `min`, `max`, or `normal`. The value defaults to `normal`.
- 740 • If the `ovf:bound` value is specified as either `min` or `max`, the item is used to specify the upper
741 or lower bound for one or more values for a given `InstanceID`. Such an item is called a range
742 marker.

743 The semantics of range markers are as follows:

- 744 • `InstanceID` and `ResourceType` shall be specified, and the `ResourceType` shall match
745 other `Item` elements with the same `InstanceID`.
- 746 • No more than one `min` range marker and no more than one `max` range marker for a given
747 `RASD`, `EPASD`, or `SASD` (identified with `InstanceID`) shall be specified.
- 748 • An `Item`, `EthernetPortItem`, or `StorageItem` element with a range marker shall have
749 a corresponding `Item`, `EthernetPortItem`, or `StorageItem` element without a range
750 marker, that is, an `Item`, `EthernetPortItem`, and `StorageItem` element with no
751 `ovf:bound` attribute or `ovf:bound` attribute with value `normal`. This corresponding item
752 specifies the default value.
- 753 • For an `Item`, `EthernetPortItem`, and `StorageItem` element where only a `min` range
754 marker is specified, the `max` value is unbounded upwards within the set of valid values for the
755 property.
- 756 • For an `Item`, `EthernetPortItem`, and `StorageItem` where only a `max` range marker is
757 specified, the `min` value is unbounded downwards within the set of valid values for the property.
- 758 • The default value shall be inside the range.
- 759 • Non-integer elements shall not be used in the range markers for `RASD`, `EPASD`, or `SASD`.

760 See D.9 for an example.

761

762 **9 Core Metadata Sections in version 2**763 Table 5 shows the core metadata sections that are defined in the `ovf` namespace.764 **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
<code>EncryptionSection</code>	Envelope	Zero or one

Section element	Parent element	Multiplicity
Specifies encryption scheme for encrypting parts of an OVF descriptor or files that it refers to.		

765 The following subclauses describe the semantics of the core sections and provide some examples. The
 766 sections are used in several places of an OVF envelope; the description of each section defines where it
 767 may be used. See the DSP8023 schema for a detailed specification of all attributes and elements.

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

770 9.1 DiskSection

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

774 The virtual disks in the `DiskSection` element may be referenced by one or more virtual systems.
 775 However, as seen from the guest software each virtual system gets individual private disks. Any level of
 776 sharing done at runtime is virtualization platform specific and not visible to the guest software. See clause
 777 9.13 for details on how to configure sharing of a virtual disk at runtime with concurrent access. See D.10
 778 for an example.

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

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

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

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

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

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

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

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

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

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

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

820 9.2 NetworkSection

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

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

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

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

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

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

836 9.3 ResourceAllocationSection

837 The `ResourceAllocationSection` element describes all resource allocation requirements of a
838 `VirtualSystemCollection` entity and applies only to the direct child `VirtualSystem` elements that
839 do not contain a `VirtualHardwareSection` element. It does not apply to a child
840 `VirtualSystemCollection` elements.

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

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

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

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

847 9.4 AnnotationSection

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

850 The `AnnotationSection` element is a valid element for the `VirtualSystem` and the
851 `VirtualSystemCollection` entities.

852 See clause 10 for details on how to localize the `Annotation` element.

853 9.5 ProductSection

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

856 Zero or more `e` elements may be specified within a `VirtualSystem` element or
857 `VirtualSystemCollection` element.

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

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

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

868 See D.14 for an example.

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

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

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

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

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

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

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

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

879 9.5.1 Property Elements

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

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

884 The `Property` elements may be grouped by using `Category` elements. The set of `Property` elements
885 grouped by a `Category` element is the sequence of `Property` elements following the `Category`
886 element, until but not including an element that is not a `Property` element. For OVF packages
887 containing a large number of `Property` elements, this may provide a simpler installation experience.
888 Similarly, each `Property` element may have a short label defined by its `Label` child element in addition
889 to a description defined by its `Description` child element. See clause 10 for details on how to localize
the `Category` element and the `Description` and `Label` child elements of the `Property` element.

891 Each `Property` element in a `ProductSection` shall be given an identifier that is unique within the
892 `ProductSection` using the `ovf:key` attribute.

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

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

899 The optional attribute `ovf:userConfigurable` determines whether the property value is configurable
900 during the installation phase. If `ovf:userConfigurable` is `FALSE` or omitted, the `ovf:value` attribute
901 specifies the value to be used for that customization parameter during installation. If
902 `ovf:userConfigurable` is `TRUE`, the `ovf:value` attribute specifies a default value for that
903 customization parameter, which may be changed during installation.

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

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

914 The `id` and the value of the `property` elements are exposed to the guest software using the OVF
915 environment file, as described in clause 9.5.1. The value of the `ovfenv:key` attribute of a `Property`
916 element exposed in the OVF environment shall be constructed from the value of the `ovf:key` attribute of
917 the corresponding `Property` element defined in a `ProductSection` entity of an OVF descriptor as
918 follows:

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

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

- 921 • `class-value` is the value of the `ovf:class` attribute of the `Property` element defined in the
 922 `ProductSection` entity. The production `[class-value "."]` shall be present if and only if
 923 `class-value` is not the empty string.
- 924 • `key-value-prod` is the value of the `ovf:key` attribute of the `Property` element defined in the
 925 `ProductSection` entity.
- 926 • `instance-value` is the value of the `ovf:instance` attribute of the `Property` element defined in
 927 the `ProductSection` entity. The production `[". " instance-value]` shall be present if and only
 928 if `instance-value` is not the empty string.

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

932 `Property` elements specified on a `VirtualSystemCollection` element are also seen by its
 933 immediate child elements (see clause 9.5.1). Child elements may refer to the properties of a parent
 934 `VirtualSystemCollection` element using macros on the form `${name}` as value for `ovf:value`
 935 attributes.

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

939

Table 6 – Property Types

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

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

944

Table 7 – Property Qualifiers

Property Type	Property Qualifier
<code>String</code>	<code>MinLen (min)</code> <code>MaxLen (max)</code> <code>ValueMap{...}</code>

uint8 sint8 uint16 sint16 uint32 sint32 uint64 sint64	ValueMap{...}
--	---------------

945 **9.6 EulaSection**

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

950 See clause 10 for details on how to localize the `License` element.

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

953 **9.7 StartupSection**

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

957 Each `VirtualSystem` element or `VirtualSystemCollection` element that is a direct child of a
 958 `VirtualSystemCollection` element may have a corresponding `Item` element in the `StartupSection`
 959 element. Note that `Item` elements may correspond to both `VirtualSystem` and `VirtualSystemCollection`
 960 entities.

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

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

- 968 • `ovf:id` shall match the value of the `ovf:id` attribute of a `Content` element which is a direct child of
 969 this `VirtualSystemCollection`. That `Content` element describes the virtual system or virtual system
 970 collection to which the actions defined in the `Item` element apply.
- 971 • `ovf:order` specifies the startup order of the item using non-negative integer values. If the `ovf:order`
 972 `= "0"` then the order is not specified. If the `ovf:order` is non-zero then the order of execution of the start action
 973 shall be the numerical ascending order of the values. The `Items` with same order identifier may be started
 974 concurrently.

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

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

- 979 • `ovf:shutdownorder` specifies the shutdown order using non-negative integer values. If the
980 `ovf:shutdownorder = "0"` then the shutdown order is not specified. If the `ovf: shutdownorder` is non-zero
981 then the order of execution of the stop action shall be the numerical descending order of the values. The
982 items with same order identifier may be stopped concurrently.
 - 983 • `ovf:startDelay` specifies a delay in seconds to wait until proceeding to the next virtual system in
984 the start sequence. The default value is 0.
 - 985 • `ovf:waitingForGuest` enables the virtualization platform to resume the startup sequence after the
986 guest software has reported it is ready. The interpretation of this is virtualization platform specific. The
987 default value is FALSE.
 - 988 • `ovf:startAction` specifies the start action to use. Valid values are `powerOn` and `none`. The default
989 value is `powerOn`.
 - 990 • `ovf:stopDelay` specifies a delay in seconds to wait until proceeding to the previous order in the
991 stop sequence. The default value is 0.
 - 992 • `ovf:stopAction` specifies the stop action to use. Valid values are `powerOff`, `guestShutdown`, and
993 `none`. The interpretation of `guestShutdown` is virtualization platform specific. The default value is
994 `powerOff`.
- 995 If the `StartupSection` element is not specified then an `ovf:order="0"` attribute is implied.

996 9.8 DeploymentOptionSection

997 The `DeploymentOptionSection` element specifies a discrete set of intended resource configurations.
998 The author of an OVF package can include sizing metadata for different configurations. The deployment
999 shall select one of the configurations, e.g., by prompting the user. The selected configuration shall be
1000 available in the OVF environment file. See ANNEX F.

1001 The `DeploymentOptionSection` specifies an ID, label, and description for each configuration. See
1002 D.17 for an example.

1003 The `DeploymentOptionSection` has the following semantics:

- 1004 • If present, the `DeploymentOptionSection` is valid only as a direct child element of the root
1005 element. Only one `DeploymentOptionSection` section shall be present in an OVF
1006 descriptor.
- 1007 • The discrete set of configurations is described with `Configuration` elements, which shall
1008 have identifiers specified by the `ovf:id` attribute that are unique in the OVF package.
- 1009 • A default `Configuration` element may be specified with the optional `ovf:default` attribute.
1010 Only one default `Configuration` element shall be specified. If no default is specified, the first
1011 element in the list is the default.
- 1012 • The `Label` and `Description` elements are localizable using the `ovf:msgid` attribute. See
1013 clause 10 for more details on internationalization support.

1014 Configurations may be used to control resources for virtual hardware and for virtual system collections.
1015 The `Item`, `EthernetPortItem`, and `StorageItem` elements in `VirtualHardwareSection`
1016 elements describe resources for `VirtualSystem` entities, while the `Item`, `EthernetPortItem`, and
1017 `StorageItem` elements in `ResourceAllocationSection` elements describe resources for virtual
1018 system collections. For these two `Item`, `EthernetPortItem`, or `StorageItem` types, the following
1019 additional semantics are defined:

- 1020 • Each `Item`, `EthernetPortItem`, and `StorageItem` has an optional
1021 `ovf:configuration` attribute, containing a list of configurations separated by a single space

1022 character. If not specified, the item shall be selected for any configuration. If specified, the item
 1023 shall be selected only if the chosen configuration ID is in the list. A configuration attribute shall
 1024 not contain a configuration ID that is not specified in the `DeploymentOptionSection`.

- 1025 • Within a single `VirtualHardwareSection` or `ResourceAllocationSection`, multiple
 1026 `Item`, `EthernetPortItem`, and `StorageItem` elements are allowed to refer to the same
 1027 `InstanceID`. A single combined `Item`, `EthernetPortItem`, or `StorageItem` for the
 1028 given `InstanceID` shall be constructed by picking up the child elements of each `Item`,
 1029 `EthernetPortItem`, or `StorageItem` element, with child elements of a former `Item`,
 1030 `EthernetPortItem`, or `StorageItem` element in the OVF descriptor not being picked up
 1031 if there is a like-named child element in a latter `Item`, `EthernetPortItem`, or
 1032 `StorageItem` element. Any attributes specified on child elements of `Item`,
 1033 `EthernetPortItem`, or `StorageItem` elements that are not picked up that way, are not
 1034 part of the combined `Item`, `EthernetPortItem`, or `StorageItem` element.
- 1035 • All `Item`, `EthernetPortItem`, `StorageItem` elements shall specify `ResourceType`, and
 1036 `Item`, `EthernetPortItem`, and `StorageItem` elements with the same `InstanceID` shall
 1037 agree on `ResourceType`.

1038 Note that the attributes `ovf:configuration` and `ovf:bound` on `Item` may be used in combination to
 1039 provide flexible configuration options.

1040 Configurations can further be used to control default values for properties and whether properties are
 1041 user configurable. For `Property` elements inside a `ProductSection`, the following additional semantic
 1042 is defined:

- 1043 • It is possible to specify alternative default property values for different configurations in a
 1044 `DeploymentOptionSection`. In addition to a `Label` and `Description` element, each
 1045 `Property` element may optionally contain `Value` elements. The `Value` element shall have
 1046 an `ovf:value` attribute specifying the alternative default and an `ovf:configuration`
 1047 attribute specifying the configuration in which this new default value should be used. Multiple
 1048 `Value` elements shall not refer to the same configuration.
- 1049 • A `Property` element may optionally have an `ovf:configuration` attribute specifying the
 1050 configuration in which this property should be user configurable. The value of
 1051 `ovf:userConfigurable` is implicitly set to `FALSE` for all other configurations, in which
 1052 case the default value of the property may not be changed during installation.

1053 9.9 OperatingSystemSection

1054 An `OperatingSystemSection` specifies the operating system installed on a virtual system. See D.18
 1055 for an example.

1056 The values for `ovf:id` should be taken from the `ValueMap` of the `CIM_OperatingSystem.OsType`
 1057 property. The description should be taken from the corresponding `Values` of the
 1058 `CIM_OperatingSystem.OsType` property.

1059 The `OperatingSystemSection` element is a valid section for a `VirtualSystem` element only.

1060 9.10 InstallSection

1061 The `InstallSection` element, if specified, indicates that the virtual system needs to be booted once in
 1062 order to install and/or configure the guest software. The guest software is expected to access the OVF
 1063 environment during that boot, and to shut down after having completed the installation and/or
 1064 configuration of the software, powering off the guest.

1065 If the `InstallSection` is not specified, this indicates that the virtual systemsystem does not need to be
1066 powered on to complete installation of guest software. See D.19 for an example.

1067 The `InstallSection` element shall be valid only for a `VirtualSystem` element.

1068 The `ovf:initialBootStopDelay` attribute specifies a delay in seconds to wait for the virtual
1069 systemsystem to power off.

1070 virtual systemIf the delay expires and the virtual systemsystem has not powered off, the deployment
1071 function shall indicate a failure.

1072 An `ovf:initialBootStopDelay` attribute value of zero indicates that the boot stop delay is not
1073 specified.

1074 Note that the guest software in the virtual system can do multiple reboots before powering off.

1075 Several virtual systems in a virtual system collection may have an `InstallSection` element defined, in
1076 which case the above step is done for each virtual system that has an `InstallSection` element.-.

1077 9.11 EnvironmentFilesSection

1078 The `EnvironmentFilesSection` enables the OVF package to specify additional environment file(s)
1079 (AEF) besides the virtual disks. These AEF's enable increased flexibility in image customization outside of
1080 virtual disk capture, allowing an OVF package to provide customized solutions by combining existing
1081 virtual disks without modifying them.

1082 The AEF contents are neither generated nor validated by the deployment function.

1083 The AEF's are included in the transport media generated by the deployment function.

1084 The AEF's are conveyed to the guest software using the indicated transport media type. The AEF's and
1085 OVF environment files are intended to use same transport media and transport media type

1086 The `EnvironmentFilesSection` shall contain a `File` element with the attributes `ovf:fileRef` and
1087 `ovf:path` for each AEF provided to the guest software.

1088 The `ovf:fileRef` attribute shall specify an existing `File` element in the `References` element. The
1089 `File` element is identified by matching its `ovf:id` attribute value with the `ovf:fileRef` attribute value.

1090 The `ovf:path` attribute specifies the relative location in the transport media (see clause 11.1) where the
1091 file should be placed, using the syntax of relative-path references in [RFC3986](#).

1092 The referenced `File` element in the `References` element identifies the content using one of the URL
1093 schemes "file", "http", or "https". For the "file" scheme, the content is static and included in
1094 the OVF package. See ANNEX F for deployment considerations

1095 For details about transport media type, see clause 11.2.

1096 9.12 BootDeviceSection

1097 Individual virtual systems use the default device boot order provided by the virtualization platform's virtual
1098 BIOS. The `BootDeviceSection` allows the OVF package author to specify particular boot
1099 configurations and boot order settings. This enables booting from non-default devices such as a NIC
1100 using PXE, a USB device or a secondary disk. Moreover there could be multiple boot configurations with
1101 different boot orders. For example, a virtual disk may need to be patched before it is bootable and a patch
1102 ISO image could be included in the OVF package.

1103 The Common Information Model (CIM) defines artifacts to deal with boot order use cases prevalent in the
1104 industry for BIOSes found in desktops and servers. The boot configuration is defined by the class
1105 `CIM_BootConfigSetting` that in turn contains one or more `CIM_BootSourceSetting` classes as
1106 defined in the CIM schema. Each class representing the boot source in turn has either the specific device
1107 or a “device type” such as disk or CD/DVD as a boot source.

1108 In the context of this specification, the `InstanceID` property of `CIM_BootSourceSetting` is used for
1109 identifying a specific device as the boot source. The `InstanceID` property of the device as specified in
1110 the `Item` description of the device in the `VirtualHardwareSection` element is used to specify the
1111 device as a boot source. In case the source is desired to be a device type, the
1112 `StructuredBootString` field is used to denote the type of device with values defined by the CIM boot
1113 control profile. See ANNEX F for deployment considerations.

1114 See D.21 for an example.

1115 **9.13 SharedDiskSection**

1116 The existing `DiskSection` element in clause 9.1 describes virtual disks in the OVF package. Virtual
1117 disks in the `DiskSection` element can be referenced by multiple virtual systems, but seen from the
1118 guest software each virtual system gets individual private disks. Any level of sharing done at runtime is
1119 deployment platform specific and not visible to the guest software.

1120 Certain applications such as clustered databases rely on multiple virtual systems sharing the same virtual
1121 disk at runtime. `SharedDiskSection` allows the OVF package to specify `Disk` elements shared by
1122 more than one virtual system at runtime, these could be virtual disks backed by an external `File`
1123 reference, or empty virtual disks without backing. It is recommended that the guest software use cluster-
1124 aware file system technology to be able to handle concurrent access. See D.22 for an example.

1125 The `SharedDiskSection` is a valid section at the outermost envelope level only.

1126 Each virtual disk is represented by a `SharedDisk` element that shall be given an identifier using the
1127 `ovf:diskId` attribute; the identifier shall be unique within the combined content of `DiskSection` and
1128 `SharedDiskSection` element. The `SharedDisk` element has the same structure as the `Disk` element
1129 in the `DiskSection` element, with the addition of an `ovf:readOnly` attribute. The `ovf:readOnly` is
1130 optional and states whether shared disk access is read-write i.e. FALSE, or read-only i.e., TRUE.

1131 Shared virtual disks are referenced from virtual hardware using the `HostResource` element as described
1132 in clause 8.3.

1133 It is optional for the virtualization platform to support the `SharedDiskSection` element. The platform
1134 should give an appropriate error message based on the value of the `ovf:required` attribute on the
1135 `SharedDiskSection` element.

1136 **9.14 ScaleOutSection**

1137 The number of virtual systems or collections of virtual system contained in an OVF package is fixed and
1138 determined by the structure inside the `Envelope` element. The `ScaleOutSection` element allows a
1139 `VirtualSystemCollection` element to contain a set of children that are homogeneous with respect to
1140 a prototypical `VirtualSystem` or `VirtualSystemCollection` element. The `ScaleOutSection`
1141 element shall cause the deployment function to replicate the prototype a number of times, thus allowing
1142 the number of instantiated virtual systems to be configured dynamically at deployment time. See for an
1143 example.

1144 This mechanism enables scaling of virtual system instances at deployment time. Scaling at runtime is not
1145 within the scope of this specification.

- 1146 The `ScaleOutSection` element is a valid section inside `VirtualSystemCollection` element only.
- 1147 The `ovf:id` attribute on `ScaleOutSection` element identifies the virtual system or collection of
1148 virtual systems prototype to be replicated.
- 1149 For the `InstanceCount` element, the `ovf:minimum` and `ovf:maximum` attribute values shall be non-
1150 negative integers and `ovf:minimum` shall be less than or equal to the value of `ovf:maximum`. The
1151 `ovf:minimum` value may be zero in which case the `VirtualSystemCollection` may contain zero instances
1152 of the prototype. If the `ovf:minimum` attribute is not present, it shall be assumed to have a value of one.
1153 If the `ovf:maximum` attribute is not present, it shall be assumed to have a value of unbounded. The
1154 `ovf:default` attribute is required and shall contain a value within the range defined by `ovf:minimum`
1155 and `ovf:maximum`.
- 1156 Each replicated instance shall be assigned a unique `ovf:id` value within the
1157 `VirtualSystemCollection` element. The unique `ovf:id` value shall be constructed from the
1158 prototype `ovf:id` value with a sequence number appended as follows:
- ```
1159 replica-ovf-id = prototype-ovf-id "-" decimal-number
1160 decimal-number = decimal-digit | (decimal-digit decimal-number)
1161 decimal-digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
```
- 1162 The syntax definitions above use ABNF with the exceptions listed in ANNEX A. The first replica shall  
1163 have sequence number one and following sequence numbers shall be incremented by one for each  
1164 replica. Note that after deployment, no `VirtualSystem` will have the prototype `ovf:id` value itself.
- 1165 If the prototype being replicated has a starting order in the `StartupSection`, all replicas shall share this  
1166 value. It is not possible to specify a particular starting sequence among replicas.
- 1167 Property values for Property elements in the prototype are prompted for once per replica created. If the  
1168 OVF package author requires a property value to be shared among instances, that Property may be  
1169 declared at the containing `VirtualSystemCollection` level and referenced by replicas as described in  
1170 clause 9.5.
- 1171 Configurations from the `DeploymentOptionSection` element may be used to control values for  
1172 `InstanceCount` element. The `InstanceCount` element may have an `ovf:configuration` attribute  
1173 specifying the configuration in which this element should be used. Multiple elements shall not refer to the  
1174 same configuration, and a configuration attribute shall not contain an `ovf:id` value that is not specified in  
1175 the `DeploymentOptionSection`. See D.23 for an example.
- 1176 **9.15 PlacementGroupSection and PlacementSection**
- 1177 Guest software may require the deployment of virtual systems with specific proximity needs. There are  
1178 two use cases:
- 1179 1) The ability to specify that two or more virtual systems should be deployed closely together because  
1180 they rely on fast communication or have a common dependency.
- 1181 2) The ability to specify that two or more virtual systems should be deployed on different platforms or  
1182 locations because of high-availability or disaster recovery considerations.
- 1183 The `PlacementGroupSection` element allows an OVF package to define a placement policy for a group of  
1184 `VirtualSystems`. The `PlacementSection` element allows the annotation of the elements with  
1185 membership of a particular placement policy group.
- 1186 Zero or more `PlacementGroupSections` may be defined at the Envelope level. The `PlacementSection`  
1187 element may be declared at the `VirtualSystem` or `VirtualSystemCollection` level.

1188 Declaring a VirtualSystemCollection a member of a placement policy group applies transitively to all child  
 1189 VirtualSystem and child Virtual System Collections elements provided that no placement policies are  
 1190 specified for the child VirtualSystem or VirtualSystemCollection.

1191 If a parent VirtualSystemCollection and child VirtualSystem(s) and/or VirtualSystemCollection(s) both  
 1192 have placement policies, the placement policies of the child VirtualSystems and/or child  
 1193 VirtualSystemCollections should be applied first. Then placement policy of the parent  
 1194 VirtualSystemCollection should be applied.

1195 In the event that there is a conflict in the placement policy the availability policy should override the affinity  
 1196 policy

1197 The ovf:id attribute in PlacementGroupSection is used to identify a placement policy. The value of the  
 1198 ovf:id attribute shall be unique within the OVF package.

1199 Placement policy group membership is specified using the ovf:group attribute in the PlacementSection.  
 1200 The value of the ovf:group attribute shall match the value of an ovf:id attribute in a  
 1201 PlacementGroupSection. The value of the ovf:group attribute shall be a comma separated text string of  
 1202 placement policy attributes.

1203 This standard defines the placement policies "affinity" and "availability", specified with the required  
 1204 ovf:policy attribute on PlacementGroupSection.

1205 The set of attributes used for availability and affinity are defined in Table 8 and Table 9.

1206 **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.           |

1207 **Table 9 – Core Sections**

1208 Table y – 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 & high bandwidth network connectivity |
|-------------------------------------------------------------------------|

1209 The placement policies that can be declared within a PlacementGroupSection are combinations of the  
 1210 availability and affinity attributes defined in Table x and Table y. The placement policy is a single string  
 1211 represented by concatenating the valid placement policy combinations using commas as separators.  
 1212 Allowed combinations of affinity and availability attributes is defined in Table 10.

1213 **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            |

1214 The availability of the parent shall be higher availability order than the availability of the child.

1215 If the placement policy is 'availability' without scoping then no availability order is specified.

1216 See D.24 for an example.

## 1217 9.16 Encryption Section

1218 It is desirable for licensing and other reasons to have an encryption scheme enabling free exchange of  
 1219 OVF appliances while ensuring that only the intended parties can use them. The XML Encryption Syntax  
 1220 and Processing standard is utilized to encrypt either the files in the reference section or any parts of the  
 1221 XML markup of an OVF document.

1222 The various aspects of OVF encryption are as shown below:

- 1223 1. block encryption  
 1224 The OVF package shall utilize block encryption algorithms as specified in the XML encryption 1.1  
 1225 documents (ref) for this purpose.
- 1226 2. key derivation  
 1227 The OVF package may use the appropriate key for this purpose. If the key is derived using a  
 1228 passphrase then the author shall use one of the key derivations specified in the XML Encryption  
 1229 1.1 standard.
- 1230 3. key transport.  
 1231 If the encryption key is embedded in the OVF package, the specified key transport mechanisms  
 1232 shall be used.

1233 This standard defines a section called the EncryptionSection as a focal point for the encryption  
 1234 functionality. This section provides a single location for placing the encryption algorithm related markup  
 1235 and the corresponding reference list to point to the OVF content that has been encrypted.

1236 Note that depending on the parts of the OVF package that has been encrypted, an OVF descriptor may  
 1237 not validate against the DSP8023 until decrypted. See D.25 for an example.

## 1238 10 Internationalization

1239 The following elements support localizable messages using the optional `ovf:msgid` attribute:

- 1240 • `Info` element on `Content`
- 1241 • `Name` element on `Content`
- 1242 • `Info` element on `Section`
- 1243 • `Annotation` element on `AnnotationSection`
- 1244 • `License` element on `EulaSection`
- 1245 • `Description` element on `NetworkSection`
- 1246 • `Description` element on `OperatingSystemSection`
- 1247 • `Description`, `Product`, `Vendor`, `Label`, and `Category` elements on `ProductSection`
- 1248 • `Description` and `Label` elements on `Property`
- 1249 • `Description` and `Label` elements on `DeploymentOptionSection`
- 1250 • `ElementName`, `Caption` and `Description` subelements on the `System` element in
- 1251 `VirtualHardwareSection`
- 1252 • `ElementName`, `Caption` and `Description` subelements on `Item` elements in
- 1253 `VirtualHardwareSection`
- 1254 • `ElementName`, `Caption` and `Description` subelements on `Item` elements in
- 1255 `ResourceAllocationSection`

1256 The `ovf:msgid` attribute contains an identifier that refers to a message that may have different values in  
1257 different locales. See D.26 for an example.

1258 The `xml:lang` attribute on the `Envelope` element shall specify the default locale for messages in the  
1259 descriptor. The attribute is optional with a default value of "en-US".

### 1260 10.1 Internal Resource Bundles

1261 Message resource bundles can be internal or external to the OVF descriptor. Internal resource bundles  
1262 are represented as `Strings` elements at the end of the `Envelope` element. See D.26 for an example.

### 1263 10.2 External Resource Bundles

1264 External resource bundles shall be listed first in the `References` section and referred to from `Strings`  
1265 elements. An external message bundle follows the same schema as the embedded one. Exactly one  
1266 `Strings` element shall be present in an external message bundle, and that `Strings` element shall not  
1267 have an `ovf:fileRef` attribute specified. See D.26 for an example.

1268 The embedded and external `Strings` elements may be interleaved, but they shall be placed at the end  
1269 of the `Envelope` element. If multiple occurrences of a `msgid` attribute with a given locale occur, a latter  
1270 value overwrites a former.

### 1271 10.3 Message Content in External File

1272 The content of all localizable messages may be stored in an external file using the optional  
1273 `ovf:fileRef` attribute on the `Msg` element. For the `License` element on `EulaSection` in particular,  
1274 this allows inclusion of a standard license text file in unaltered form without any XML header or footer.

- 1275 The `ovf:fileRef` attribute denotes the message content by identifying an existing `File` element in the  
1276 `References` element, the `File` element is identified by matching its `ovf:id` attribute value with the  
1277 `ovf:fileRef` attribute value. The content of an external file referenced using `ovf:fileRef` shall be  
1278 interpreted as plain text in UTF-8 Unicode.
- 1279 If the referenced file is not available, the embedded content of the `Msg` element shall be used.
- 1280 The optional `ovf:fileRef` attribute may appear on `Msg` elements in both internal and external `Strings`  
1281 resource bundles. See D.27 for an example.

## 1282 11 OVF Environment and OVF Environment File

- 1283 The OVF environment defines how the guest software and the virtualization platform interact. The OVF  
1284 environment enables the guest software to access information about the virtualization platform, such as  
1285 the user-specified values for the properties defined in the OVF descriptor.
- 1286 The `dsp8027_1.1.0.xsd` XML schema definition file for the OVF environment contains the elements  
1287 and attributes that define the format and semantics of an XML document which constituent OVF  
1288 environment file (OEF). The OEF shall validate with the `DSP8027_1.1.0.xsd`
- 1289 The OEF is created on a per virtual system basis by the deployment function. The basis of the OEF is the  
1290 OVF descriptor, OVF operational metadata, OVF property values, policy metadata, and other user  
1291 provided values.
- 1292 OEF is provided to the guest software about the environment in which it is being executed. The way that  
1293 the OEF is obtained depends on the transport media type. See D.28 for an example.
- 1294 The value of the `ovfenv:id` attribute of the `Environment` element shall match the value of the `ovf:id`  
1295 attribute of the `VirtualSystem` entity describing this virtual system.
- 1296 The `PlatformSection` element contains optional information provided by the deployment function. The  
1297 `Kind`, `Version`, and `Vendor` elements describe the virtualization platform. The `Locale` and `TimeZone`  
1298 elements describe the current locale and time zone.
- 1299 The `PropertySection` element contains `Property` elements with key/value pairs corresponding to all  
1300 properties specified in the OVF descriptor for the current virtual system, as well as properties specified for  
1301 the immediate parent `VirtualSystemCollection`, if one exists. The environment presents properties  
1302 as a simple list to make it easy for applications to parse. Furthermore, the single list format supports the  
1303 override semantics where a property on a `VirtualSystem` may override one defined on a parent  
1304 `VirtualSystemCollection`. The overridden property shall not be in the list. Overriding may occur if a  
1305 property in the current virtual system and a property in the parent `VirtualSystemCollection` has  
1306 identical `ovf:key`, `ovf:class`, and `ovf:instance` attribute values; see 9.5. In this case, the value of  
1307 an overridden parent property may be obtained by adding a differently named child property referencing  
1308 the parent property with a macro; see 9.5.
- 1309 An `Entity` element shall exist for each sibling `VirtualSystem` and `VirtualSystemCollection`, if  
1310 any are present. The value of the `ovfenv:id` attribute of the `Entity` element shall match the value of  
1311 the `ovf:id` attribute of the sibling entity. The `Entity` elements contain the property key/value pairs in  
1312 the sibling's OVF environment documents, so the content of an `Entity` element for a particular sibling  
1313 shall contain the exact `PropertySection` seen by that sibling. This information can be used, for  
1314 example, to make configuration information such as IP addresses available to `VirtualSystems` being  
1315 part of a multi-tiered application.
- 1316 Table 11 shows the core sections that are defined.

1317

**Table 11 – Core Sections**

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

1318 The e is extensible by providing new section types. The deployment function should ignore unknown  
 1319 section types and elements specified in OEF.

1320 **11.1 Transport Media**

1321 The transport media refers to the format in which information is conveyed to the guest software. The  
 1322 transport media (e.g. ISO image) is generated by the deployment function that includes OVF environment  
 1323 file and any additional environment file(s).

1324 If the transport media type is 'iso' the generated ISO image shall comply with the ISO 9660 specification  
 1325 with support for Joliet extensions.

1326 The transport media shall contain the OVF environment file for this particular virtual system shall be  
 1327 presented in an XML file named `ovf-env.xml` that is contained in the root directory of the transport  
 1328 media. The guest software can now access the information.

1329 For additional environment file (s), the transport media shall have the root location relative to the  
 1330 `ovf:path` attribute shall be in a directory named "ovfiles" contained in the root directory . This provides  
 1331 an access mechanism for the guest software.

1332 Other custom transport media may support this mechanism. Custom transport medium shall specify how  
 1333 to access multiple data sources from a root location. See D.20 for an example. The access mechanism  
 1334 for the guest software is not specified.

1335 **11.2 Transport Media Type**

1336 The transport media type refers to a mechanism to convey transport media over any data link or  
 1337 removable storage medium (i.e. CD/DVD-ROM) from deployment functions to guest software.

1338 The `iso` transport media type shall support this mechanism, see clause 11.2 for details.

1339 This standard defines the "iso" transport type to meet the need for interoperability.

1340 The transport media can be communicated in a number of ways to the guest software. These ways are  
 1341 called transport media types. The transport media types are specified in the OVF descriptor by the  
 1342 `ovf:transport` attribute of `VirtualHardwareSection`. Several transport media types may be  
 1343 specified, separated by a single space character, in which case an implementation is free to use any of  
 1344 them.

1345 To enable interoperability, this specification defines an "iso" transport media type which all  
 1346 implementations that support CD-ROM devices are required to support. The `iso` transport media type  
 1347 communicates the environment document by making a dynamically generated ISO image available to the  
 1348 guest software.

1349 To support the `iso` transport media type, prior to booting a virtual system, an implementation shall make  
 1350 an ISO read-only disk image available as backing for a disconnected CD-ROM. If the `iso` transport

- 1351 media type is selected for a `VirtualHardwareSection`, at least one disconnected CD-ROM device  
1352 shall be present in this section.
- 1353 If the virtual system prior to booting had more than one disconnected CD-ROM, the guest software may  
1354 have to scan connected CD-ROM devices in order to locate the ISO image containing the `ovf-env.xml`  
1355 file.
- 1356 The transport media containing the OVF environment file shall be made available to the guest software  
1357 on every boot of the virtual system.
- 1358 Support for the "iso" transport media type is not a requirement for virtual hardware architectures or  
1359 guest software which do not have CD-ROM device support.
- 1360 To be conformant with this specification, any transport media type other than `iso` shall be given by a URI  
1361 that identifies an unencumbered specification on how to use the transport media type. The specification  
1362 need not be machine readable, but it shall be static and unique so that it may be used as a key by  
1363 software reading an OVF descriptor to uniquely determine the transport media type. The specification  
1364 shall be sufficient for a skilled person to properly interpret the transport media type mechanism for  
1365 implementing the protocols. The URIs should be resolvable.
- 1366



## ANNEX A (informative)

1367  
1368  
1369  
1370

### Symbols and Conventions

1371 XML examples use the XML namespace prefixes defined in Table 1. The XML examples use a style to  
1372 not specify namespace prefixes on child elements. Note that XML rules define that child elements  
1373 specified without namespace prefix are from the namespace of the parent element, and not from the  
1374 default namespace of the XML document. Throughout the document, whitespace within XML element  
1375 values is used for readability. In practice, a service can accept and strip leading and trailing whitespace  
1376 within element values as if whitespace had not been used.

1377 Syntax definitions in this document use Augmented BNF (ABNF) as defined in IETF [RFC5234](#) with the  
1378 following exceptions:

- 1379 • Rules separated by a bar (|) represent choices, instead of using a forward slash (/) as defined in  
1380 ABNF.
- 1381 • Any characters must be processed case sensitively, instead of case-insensitively as defined in  
1382 ABNF.
- 1383 • Whitespace (i.e., the space character U+0020 and the tab character U+0009) is allowed between  
1384 syntactical elements, instead of assembling elements without whitespace as defined in ABNF.

1385

## ANNEX B (normative)

### OVF XSD

1386  
1387  
1388  
1389

1390 Normative copies of the XML schemas for this specification may be retrieved by resolving the following  
1391 URLs:

1392  
1393  
1394

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

1395 Any `xs:documentation` content in XML schemas for this specification is informative and provided only  
1396 for convenience.

1397 Normative copies of the XML schemas for the WS-CIM mapping ([DSP0230](#)) of  
1398 `CIM_ResourceAllocationSystemSettingsData`, `CIM_VirtualSystemSettingData`,  
1399 `CIM_EthernetPortAllocationSettingData`, `CIM_StorageAllocationSettingData` and  
1400 `CIM_OperatingSystem`, may be retrieved by resolving the following URLs:

1401  
1402  
1403  
1404  
1405  
1406  
1407

[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)

1408 This specification is based on the following CIM MOFs:

1409  
1410  
1411  
1412  
1413  
1414

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

## ANNEX C (informative)

### OVF Mime Type Registration Template

1415 Registration Template

1420 To: ietf-types@iana.org

1421 Subject: Registration of media type Application/OVF

1422 Type name: Application

1423 Subtype name: OVF

1424 Required parameters: none

1425 Optional parameters: none

1426 Encoding considerations: binary

1427 Security considerations:

- 1428 • An OVF package contains active content that is expected to be launched in a virtual system. The  
1429 OVF standard, section 5.1 states: “An OVF package may be signed by signing the manifest file.  
1430 The digest of the manifest file is stored in a certificate file with extension .cert file along with the  
1431 base64-encoded X.509 certificate. The .cert file shall have the same base name as the .ovf file  
1432 and be a sibling of the .ovf file. A consumer of the OVF package shall verify the signature and  
1433 should validate the certificate.
- 1434 • An OVF package may contain passwords as part of the configuration information. The OVF  
1435 standard, section 9.5 states: “The optional Boolean attribute ovf:password indicates that the  
1436 property value may contain sensitive information. The default value is FALSE. OVF  
1437 implementations prompting for property values are advised to obscure these values when  
1438 ovf:password is set to TRUE. This is similar to HTML text input of type password. Note that this  
1439 mechanism affords limited security protection only. Although sensitive values are masked from  
1440 casual observers, default values in the OVF descriptor and assigned values in the OVF  
1441 environment are still passed in clear text. “

1442 Interoperability considerations:

- 1443 • OVF has demonstrated interoperability via multiple, interoperating implementations in the market.

1444 Published specification:

- 1445 • DSP0243\_2.0.0.pdf

1446 Applications that use this media type:

- 1447 • Implementations of the DMTF Standard: Cloud Infrastructure Management Interface (CIMI)  
1448 (DSP0263\_1.0.0.pdf)
- 1449 • Implementations of the SNIA Cloud Data Management Interface (CDMI) – OVF Extension

1450 Additional information:

- 1451 • Magic number(s): none
- 1452 • File extension(s): .ova
- 1453 • Macintosh file type code(s): none
- 1454 • Person & email address to contact for further information:

- 1455 • Intended usage: (One of COMMON, LIMITED USE or OBSOLETE.)
- 1456 • Restrictions on usage: (Any restrictions on where the media type can be used go here.)
- 1457 • Author:
- 1458 • Change controller:
- 1459

## ANNEX D (informative)

### OVF Examples

1460  
1461  
1462  
1463

#### D.1 Examples of OVF Package Structure

1464  
1465  
1466  
1467  
1468  
1469  
1470  
1471  
1472  
1473

##### 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
```

1474  
1475

##### EXAMPLE 2:

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

1476  
1477  
1478

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

1479

##### EXAMPLE 3:

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

1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488

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

1489  
1490

##### 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:

1491  
1492  
1493

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

1494

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

1495  
1496  
1497  
1498  
1499  
1500  
1501  
1502  
1503  
1504

```
MIIBgjCCASwCAQQwDQYJKoZIhvcNAQEEBQAwODELMAkGA1UEBhMCQVUxDDAKBgNV
BAgTA1FMRDEbMBkGA1UEAxMSU1NMZWZ5L3JzYSB0ZXN0IENBMB4XDTE1MTAwOTIz
MzIwNVowXDTk4MDCwNTIzMzIwNVowYDELMAkGA1UEBhMCQVUxDDAKBgNVBAgTA1FM
RDEZMBcGA1UEChMQTWluY29tIFB0eS4gTHRkLjELMAkGA1UECXMxMzIwNVowXDTk4
MDCwNTIzMzIwNVowYDELMAkGA1UEC3Q3BAMTElNTTGVheSBkZW1vIHN1cnZlcjBcMA0GCSqGSIb3DQEBAQUAA0sAMEgCQQC3
LCXcScWua0PFLkHBLm2VejqpA1F4RQ8q0VjRiPafjx/Z/aWH3ipdMVvuJGa/wFXb
/nDFLDlFwP+oCPwhBtVPAgMBAAEwDQYJKoZIhvcNAQEEBQADQQArNFsihWIjBzb0
DcsU0BvL2bvSwJrPEqFlkDq3F4M6EgutL9axEcANWgbbEdAvNJD1dmEmoWny27Pn
Ims6ZOZB
```

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

#### D.2 Examples of Distribution of Files

1505  
1506  
1507  
1508  
1509

##### EXAMPLE 1:

An example of an OVF package as a compressed archive:

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

1510 EXAMPLE 2:  
 1511 An example of an OVF package as a set of files on Web server follows:  
 1512 <http://mywebsite/virtualappliances/package.ovf>  
 1513 <http://mywebsite/virtualappliances/vmdisk1.vmdk>  
 1514 <http://mywebsite/virtualappliances/vmdisk2.vmdk>  
 1515 <http://mywebsite/virtualappliances/resource.iso>  
 1516 <http://mywebsite/virtualappliances/de-DE-resources.xml>

### 1517 D.3 Example of Envelope Element

1518 An example of the structure of an OVF descriptor with the top-level Envelope element  
 1519 follows:

```

1520 <?xml version="1.0" encoding="UTF-8"?>
1521 <Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1522 xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-
1523 schema/2/CIM_VirtualSystemSettingData"
1524 xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
1525 schema/2/CIM_ResourceAllocationSettingData"
1526 xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/2"
1527 xmlns="http://schemas.dmtf.org/ovf/envelope/2"
1528 xml:lang="en-US">
1529 <References>
1530 <File ovf:id="de-DE-resources.xml" ovf:size="15240"
1531 ovf:href="http://mywebsite/virtualappliances/de-DE-resources.xml"/>
1532 <File ovf:id="file1" ovf:href="vmdisk1.vmdk" ovf:size="180114671"/>
1533 <File ovf:id="file2" ovf:href="vmdisk2.vmdk" ovf:size="4882023564"
1534 ovf:chunkSize="2147483648"/>
1535 <File ovf:id="file3" ovf:href="resource.iso" ovf:size="212148764"
1536 ovf:compression="gzip"/>
1537 <File ovf:id="icon" ovf:href="icon.png" ovf:size="1360"/>
1538 </References>
1539 <!-- Describes meta-information about all virtual disks in the package -->
1540 <DiskSection>
1541 <Info>Describes the set of virtual disks</Info>
1542 <!-- Additional section content -->
1543 </DiskSection>
1544 <!-- Describes all networks used in the package -->
1545 <NetworkSection>
1546 <Info>List of logical networks used in the package</Info>
1547 <!-- Additional section content -->
1548 </NetworkSection>
1549 <SomeSection ovf:required="false">
1550 <Info>A plain-text description of the content</Info>
1551 <!-- Additional section content -->
1552 </SomeSection>
1553 <!-- Additional sections can follow -->
1554 <VirtualSystemCollection ovf:id="Some Product">
1555 <!-- Additional sections including VirtualSystem or VirtualSystemCollection-->
1556 </VirtualSystemCollection >
1557 <Strings xml:lang="de-DE">
1558 <!-- Specification of message resource bundles for de-DE locale -->
1559 </Strings>
1560 </Envelope>

```

## 1561 D.4 Example of File References

1562 EXAMPLE 1:

1563 The following example shows different types of file references:

```
1564 <File ovf:id="disk1" ovf:href="disk1.vmdk"/>
1565 <File ovf:id="disk2" ovf:href="disk2.vmdk" ovf:size="5368709120"
1566 ovf:chunkSize="2147483648"/>
1567 <File ovf:id="iso1" ovf:href="resources/image1.iso"/>
1568 <File ovf:id="iso2" ovf:href="http://mywebsite/resources/image2.iso"/>
```

1570 EXAMPLE 2:

1571 The following example shows manifest entries corresponding to the file references  
1572 above:

```
1573 SHA1(disk1.vmdk)= 3e19644ec2e806f38951789c76f43e4a0ec7e233
1574 SHA1(disk2.vmdk.000000000)= 4f7158731ff434380bf217da248d47a2478e79d8
1575 SHA1(disk2.vmdk.000000001)= 12849daeeaf43e7a89550384d26bd437bb8defaf
1576 SHA1(disk2.vmdk.000000002)= 4cdd21424bd9eeafa4c42112876217de2ee5556d
1577 SHA1(resources/image1.iso)= 72b37ff3fdd09f2a93f1b8395654649b6d06b5b3
1578 SHA1(http://mywebsite/resources/image2.iso)=
1579 d3c2d179011c970615c5cf10b30957d1c4c968ad
```

## 1580 D.5 Example of Content Element

1581 An example of a VirtualSystem element structure follows:

```
1582 <VirtualSystem ovf:id="simple-app">
1583 <Info>A virtual system</Info>
1584 <Name>Simple Appliance</Name>
1585 <SomeSection>
1586 <!-- Additional section content -->
1587 </SomeSection>
1588 <!-- Additional sections can follow -->
1589 </VirtualSystem>
```

1591 An example of a VirtualSystemCollection element structure follows:

```
1592 <VirtualSystemCollection ovf:id="multi-tier-app">
1593 <Info>A collection of virtual systems</Info>
1594 <Name>Multi-tiered Appliance</Name>
1595 <SomeSection>
1596 <!-- Additional section content -->
1597 </SomeSection>
1598 <!-- Additional sections can follow -->
1599 <VirtualSystem ovf:id="...">
1600 <!-- Additional sections -->
1601 </VirtualSystem>
1602 <!-- Additional VirtualSystem or VirtualSystemCollection elements can follow-->
1603 </VirtualSystemCollection>
```

## 1604 D.6 Examples of Extensibility

1605 EXAMPLE 1:

```
1606 <!-- Optional custom section example -->
1607 <othersns:IncidentTrackingSection ovf:required="false">
1608 <Info>Specifies information useful for incident tracking purposes</Info>
1609 <BuildSystem>Acme Corporation Official Build System</BuildSystem>
1610 <BuildNumber>102876</BuildNumber>
1611 <BuildDate>10-10-2008</BuildDate>
1612 </othersns:IncidentTrackingSection>
```

```

1613
1614 EXAMPLE 2:
1615 <!-- Open content example (extension of existing type) -->
1616 <AnnotationSection>
1617 <Info>Specifies an annotation for this virtual system</Info>
1618 <Annotation>This is an example of how a future element (Author) can still be
1619 parsed by older clients</Annotation>
1620 <!-- AnnotationSection extended with Author element -->
1621 <others:Author ovf:required="false">John Smith</others:Author>
1622 </AnnotationSection>
1623
1624 EXAMPLE 3:
1625 <!-- Optional custom attribute example -->
1626 <Network ovf:name="VS network" others:desiredCapacity="1 Gbit/s">
1627 <Description>The main network for VSs</Description>
1628 </Network>

```

## 1629 D.7 Examples of VirtualHardwareSection

```

1630 EXAMPLE 1:
1631 Example of VirtualHardwareSection:
1632 <VirtualHardwareSection>
1633 <Info>Memory = 4 GB, CPU = 1 GHz, Disk = 100 GB, 1 Ethernet nic</Info>
1634 <Item>
1635 <rasd:AllocationUnits>Hertz*10^9</rasd:AllocationUnits>
1636 <rasd:Description>Virtual CPU</rasd:Description>
1637 <rasd:ElementName>1 GHz virtual CPU</rasd:ElementName>
1638 <rasd:InstanceID>1</rasd:InstanceID>
1639 <rasd:Reservation>1</rasd:Reservation>
1640 <rasd:ResourceType>3</rasd:ResourceType>
1641 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
1642 <rasd:VirtualQuantityUnit>Count</ rasd:VirtualQuantityUnit>
1643 </Item>
1644 <Item>
1645 <rasd:AllocationUnits>byte*2^30</rasd:AllocationUnits>
1646 <rasd:Description>Memory</rasd:Description>
1647 <rasd:ElementName>1 GByte of memory</rasd:ElementName>
1648 <rasd:InstanceID>2</rasd:InstanceID>
1649 <rasd:Limit>4</rasd:Limit>
1650 <rasd:Reservation>4</rasd:Reservation>
1651 <rasd:ResourceType>4</rasd:ResourceType>
1652 </Item>
1653 <EthernetPortItem>
1654 <rasd:AllocationUnits>bit / second *2^30 </rasd:AllocationUnits>
1655 <epasd:Connection>VS Network</epasd:Connection>
1656 <epasd:Description>Virtual NIC</epasd:Description>
1657 <epasd:ElementName>Ethernet Port</epasd:ElementName>
1658 <epasd:NetworkPortProfileID>1</epasd:NetworkPortProfileID>
1659 <epasd:NetworkPortProfileIDType>4</epasd:NetworkPortProfileIDType>
1660 <epasd:ResourceType>10</epasd:ResourceType>
1661 <epasd:VirtualQuantity>1</epasd:VirtualQuantity>
1662 <epasd:VirtualQuantityUnits>Count</epasd:VirtualQuantityUnits>
1663 </EthernetPortItem>
1664 <StorageItem>
1665 <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
1666 <sasd:Description>Virtual Disk</sasd:Description>

```



```

1667 <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
1668 <sasd:Reservation>100</sasd:Reservation>
1669 <sasd:ResourceType>31</sasd:ResourceType>
1670 <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
1671 <sasd:VirtualQuantityUnit>Count</sasd:VirtualQuantityUnit>
1672 </StorageItem>
1673 </VirtualHardwareSection>
1674
1675 EXAMPLE 2:
1676 <rasd:ResourceSubType>buslogic lsilogic</rasd:ResourceSubType>

```

## 1677 D.8 Examples of Virtual Hardware Elements

```

1678 EXAMPLE 1:
1679 The following example shows a description of memory size:
1680 <Item>
1681 <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1682 <rasd:Description>Memory Size</rasd:Description>
1683 <rasd:ElementName>256 MB of memory</rasd:ElementName>
1684 <rasd:InstanceID>2</rasd:InstanceID>
1685 <rasd:ResourceType>4</rasd:ResourceType>
1686 <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
1687 </Item>
1688
1689 EXAMPLE 2:
1690 The following example shows a description of a virtual Ethernet adapter:
1691 <EthernetPortItem>
1692 <epasd:Address>00-16-8B-DB-00-5E</epasd:Address>
1693 <epasd:Connection>VS Network</epasd:Connection>
1694 <epasd:Description>Virtual NIC</epasd:Description>
1695
1696 <epasd:ElementName>Ethernet Port 1</epasd:ElementName>
1697 <epasd:InstanceID>3</epasd:InstanceID>
1698 <epasd:NetworkPortProfileID>1</epasd:NetworkPortProfileID>
1699 <epasd:NetworkPortProfileIDType>4</epasd:NetworkPortProfileIDType>
1700 <epasd:VirtualQuantityUnits>1</epasd:VirtualQuantityUnits>
1701 </EthernetPortItem>
1702
1703 EXAMPLE 3:
1704 The following example shows a description of a virtual storage:
1705 <StorageItem>
1706 <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
1707 <sasd:Description>Virtual Disk</sasd:Description>
1708 <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
1709 <sasd:InstanceID>4</sasd:InstanceID>
1710 <sasd:Reservation>100</sasd:Reservation>
1711 <sasd:ResourceType>31</sasd:ResourceType>
1712 <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
1713 </StorageItem>

```

## 1714 D.9 Example of Ranges on Elements

```

1715 EXAMPLE:
1716 The following example shows the use of range markers:
1717 <VirtualHardwareSection>
1718 <Info>...</Info>

```

```

1719 <Item>
1720 <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1721 <rasd:ElementName>512 MB memory size</rasd:ElementName>
1722 <rasd:InstanceID>0</rasd:InstanceID>
1723 <rasd:ResourceType>4</rasd:ResourceType>
1724 <rasd:VirtualQuantity>512</rasd:VirtualQuantity>
1725 </Item>
1726 <Item ovf:bound="min">
1727 <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1728 <rasd:ElementName>384 MB minimum memory size</rasd:ElementName>
1729 <rasd:InstanceID>0</rasd:InstanceID>
1730 <rasd:Reservation>384</rasd:Reservation>
1731 <rasd:ResourceType>4</rasd:ResourceType>
1732 </Item>
1733 <Item ovf:bound="max">
1734 <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1735 <rasd:ElementName>1024 MB maximum memory size</rasd:ElementName>
1736 <rasd:InstanceID>0</rasd:InstanceID>
1737 <rasd:Reservation>1024</rasd:Reservation>
1738 <rasd:ResourceType>4</rasd:ResourceType>
1739 </Item>
1740 </VirtualHardwareSection>

```

## 1741 D.10 Example of DiskSection

```

1742 EXAMPLE: The following example shows a description of virtual disks:
1743 <DiskSection>
1744 <Info>Describes the set of virtual disks</Info>
1745 <Disk ovf:diskId="vmdisk1" ovf:fileRef="file1" ovf:capacity="8589934592"
1746 ovf:populatedSize="3549324972"
1747 ovf:format=
1748 "http://www.vmware.com/interfaces/specifications/vmdk.html#sparse">
1749 </Disk>
1750 <Disk ovf:diskId="vmdisk2" ovf:capacity="536870912">
1751 </Disk>
1752 <Disk ovf:diskId="vmdisk3" ovf:capacity="${disk.size}"
1753 ovf:capacityAllocationUnits="byte * 2^30">
1754 </Disk>
1755 </DiskSection>

```

## 1756 D.11 Example of NetworkSection

```

1757 <NetworkSection>
1758 <Info>List of logical networks used in the package</Info>
1759 <Network ovf:name="VS Network">
1760 <Description>The network that the service will be available on</Description>
1761 <NetworkPortProfile>
1762 <Item>
1763 <epasd:AllocationUnits>GigaBits per Second</epasd:AllocationUnits>
1764 <epasd:ElementName>Network Port Profile 1</epasd:ElementName>
1765 <epasd:InstanceID>1</epasd:InstanceID>
1766 <epasd:NetworkPortProfileID>1</epasd:NetworkPortProfileID>
1767 <epasd:NetworkPortProfileIDType>4</epasd:NetworkPortProfileIDType>
1768 <epasd:Reservation>1</epasd:Reservation>
1769 </Item>
1770 </NetworkPortProfile>

```

```
1771 </Network>
1772 </NetworkSection>
```

## 1773 D.12 Example of ResourceAllocationSection

```
1774 <ResourceAllocationSection>
1775 <Info>Defines reservations for CPU and memory for the collection of VSs</Info>
1776 <Item>
1777 <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1778 <rasd:ElementName>300 MB reservation</rasd:ElementName>
1779 <rasd:InstanceID>0</rasd:InstanceID>
1780 <rasd:Reservation>300</rasd:Reservation>
1781 <rasd:ResourceType>4</rasd:ResourceType>
1782 </Item>
1783 <Item ovf:configuration="..." ovf:bound="...">
1784 <rasd:AllocationUnits>hertz * 10^6</rasd:AllocationUnits>
1785 <rasd:ElementName>500 MHz reservation</rasd:ElementName>
1786 <rasd:InstanceID>0</rasd:InstanceID>
1787 <rasd:Reservation>500</rasd:Reservation>
1788 <rasd:ResourceType>3</rasd:ResourceType>
1789 </Item>
1790 <EthernetPortItem>
1791 <epasd:Address>00-16-8B-DB-00-5E</epasd:Address>
1792 <epasd:Connection>VS Network</epasd:Connection>
1793 <epasd:Description>Virtual NIC</epasd:Description>
1794 <epasd:ElementName>Ethernet Port 1</epasd:ElementName>
1795 <epasd:InstanceID>3</epasd:InstanceID>
1796 <epasd:NetworkPortProfileID>1</epasd:NetworkPortProfileID>
1797 <epasd:NetworkPortProfileIDType>4</epasd:NetworkPortProfileIDType>
1798 <epasd:VirtualQuantityUnits>1</epasd:VirtualQuantityUnits>
1799 </EthernetPortItem>
1800 <StorageItem>
1801 <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
1802 <sasd:Description>Virtual Disk</sasd:Description>
1803 <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
1804 <sasd:InstanceID>4</sasd:InstanceID>
1805 <sasd:Reservation>100</sasd:Reservation>
1806 <sasd:ResourceType>31</sasd:ResourceType>
1807 <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
1808 </StorageItem>
1809 </ResourceAllocationSection>
```

## 1810 D.13 Example of Annotation

```
1811 <AnnotationSection>
1812 <Info>An annotation on this service. It can be ignored</Info>
1813 <Annotation>Contact customer support if you have any problems</Annotation>
1814 </AnnotationSection >
```

## 1815 D.14 Example of Product Section

```
1816 <ProductSection ovf:class="com.mycrm.myservice" ovf:instance="1">
1817 <Info>Describes product information for the service</Info>
1818 <Product>MyCRM Enterprise</Product>
1819 <Vendor>MyCRM Corporation</Vendor>
1820 <Version>4.5</Version>
1821 <FullVersion>4.5-b4523</FullVersion>
```

```

1822 <ProductUrl>http://www.mycrm.com/enterprise</ProductUrl>
1823 <VendorUrl>http://www.mycrm.com</VendorUrl>
1824 <Icon ovf:height="32" ovf:width="32" ovf:mimeType="image/png" ovf:fileRef="icon">
1825 <Category>Email properties</Category>
1826 <Property ovf:key="admin.email" ovf:type="string" ovf:userConfigurable="true">
1827 <Label>Admin email</Label>
1828 <Description>Email address of administrator</Description>
1829 </Property>
1830 <Category>Admin properties</Category>
1831 <Property ovf:key="app_log" ovf:type="string" ovf:value="low"
1832 ovf:userConfigurable="true">
1833 <Description>Loglevel for the service</Description>
1834 </Property>
1835 <Property ovf:key="app_isSecondary" ovf:value="false" ovf:type="boolean">
1836 <Description>Cluster setup for application server</Description>
1837 </Property>
1838 <Property ovf:key="app_ip" ovf:type="string" ovf:value="{appserver-vm}">
1839 <Description>IP address of the application server VS</Description>
1840 </Property>
1841 </ProductSection>

```

## 1842 D.15 Example of EULA Section

```

1843 <EulaSection>
1844 <Info>Licensing agreement</Info>
1845 <License>
1846 Lorem ipsum dolor sit amet, ligula suspendisse nulla pretium, rhoncus tempor placerat
1847 fermentum, enim integer ad vestibulum volutpat. Nisl rhoncus turpis est, vel elit,
1848 congue wisi enim nunc ultricies sit, magna tincidunt. Maecenas aliquam maecenas ligula
1849 nostra, accumsan taciti. Sociis mauris in integer, a dolor netus non dui aliquet,
1850 sagittis felis sodales, dolor sociis mauris, vel eu libero cras. Interdum at. Eget
1851 habitasse elementum est, ipsum purus pede porttitor class, ut adipiscing, aliquet sed
1852 auctor, imperdiet arcu per diam dapibus libero duis. Enim eros in vel, volutpat nec
1853 pellentesque leo, scelerisque.
1854 </License>
1855 </EulaSection>

```

## 1856 D.16 Example of StartupSection

```

1857 <StartupSection>
1858 <Item ovf:id="vm1" ovf:order="0" ovf:startDelay="30" ovf:stopDelay="0"
1859 ovf:startAction="powerOn" ovf:waitingForGuest="true"
1860 ovf:stopAction="powerOff"/>
1861 <Item ovf:id="teamA" ovf:order="0"/>
1862 <Item ovf:id="vm2" ovf:order="1" ovf:startDelay="0" ovf:stopDelay="20"
1863 ovf:startAction="powerOn" ovf:stopAction="guestShutdown"/>
1864 </StartupSection>

```

## 1865 D.17 Example of DeploymentOptionSection

```

1866 <DeploymentOptionSection>
1867 <Configuration ovf:id="minimal">
1868 <Label>Minimal</Label>
1869 <Description>Some description</Description>
1870 </Configuration>
1871 <Configuration ovf:id="normal" ovf:default="true">
1872 <Label>Typical</Label>

```

```

1873 <Description>Some description</Description>
1874 </Configuration>
1875 <!-- Additional configurations -->
1876 </DeploymentOptionSection>
1877
1878 EXAMPLE 1: The following example shows a VirtualHardwareSection:
1879 <VirtualHardwareSection>
1880 <Info>...</Info>
1881 <Item>
1882 <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1883 <rasd:ElementName>512 MB memory size and 256 MB reservation</rasd:ElementName>
1884 <rasd:InstanceID>0</rasd:InstanceID>
1885 <rasd:Reservation>256</rasd:Reservation>
1886 <rasd:ResourceType>4</rasd:ResourceType>
1887 <rasd:VirtualQuantity>512</rasd:VirtualQuantity>
1888 </Item>
1889 ...
1890 <Item ovf:configuration="big">
1891 <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1892 <rasd:ElementName>1024 MB memory size and 512 MB reservation</rasd:ElementName>
1893 <rasd:InstanceID>0</rasd:InstanceID>
1894 <rasd:Reservation>512</rasd:Reservation>
1895 <rasd:ResourceType>4</rasd:ResourceType>
1896 <rasd:VirtualQuantity>1024</rasd:VirtualQuantity>
1897 </Item>
1898 </VirtualHardwareSection>
1899

```

```

1900 EXAMPLE 2: The following shows an example ProductSection:
1901 <ProductSection>
1902 <Property ovf:key="app.adminEmail" ovf:type="string" ovf:userConfigurable="true"
1903 ovf:configuration="standard">
1904 <Label>Admin email</Label>
1905 <Description>Email address of service administrator</Description>
1906 </Property>
1907 <Property ovf:key="app.log" ovf:type="string" ovf:value="low"
1908 ovf:userConfigurable="true">
1909 <Label>Loglevel</Label>
1910 <Description>Loglevel for the service</Description>
1911 <Value ovf:value="none" ovf:configuration="minimal">
1912 </Property>
1913 </ProductSection>

```

1914 In the example above, the app.adminEmail property is only user configurable in the  
1915 standard configuration, while the default value for the app.log property is changed  
1916 from low to none in the minimal configuration.

## 1917 D.18 Example of OperatingSystemSection

```

1918 <OperatingSystemSection ovf:id="76">
1919 <Info>Specifies the operating system installed</Info>
1920 <Description>Microsoft Windows Server 2008</Description>
1921 </OperatingSystemSection>

```

## 1922 D.19 Example of InstallSection

```

1923 <InstallSection ovf:initialBootStopDelay="300">

```

```

1924 <Info>Specifies that the virtual system needs to be booted once after having
1925 created the guest software in order to install and/or configure the software
1926 </Info>
1927 </InstallSection>

```

## 1928 D.20 Example of EnvironmentFilesSection

1929 EXAMPLE:

```

1930 <Envelope>
1931 <References>
1932 ...
1933 <File ovf:id="config" ovf:href="config.xml" ovf:size="4332"/>
1934 <File ovf:id="resources" ovf:href="http://mywebsite/resources/resources.zip"/>
1935 </References>
1936 ...
1937 <VirtualSystem ovf:id="...">
1938 ...
1939 <ovf:EnvironmentFilesSection ovf:required="false" ovf:transport="iso">
1940 <Info>Config files to be included in OVF environment</Info>
1941 <ovf:File ovf:fileRef="config" ovf:path="setup/cfg.xml"/>
1942 <ovf:File ovf:fileRef="resources" ovf:path="setup/resources.zip"/>
1943 </ovf:EnvironmentFilesSection>
1944 ...
1945 </VirtualSystem>
1946 ...
1947 </Envelope>

```

1948 In the example above, the file config.xml in the OVF package will be copied to the OVF  
1949 environment ISO image and be accessible to the guest software in location  
1950 /ovffiles/setup/cfg.xml, while the file resources.zip will be accessible in location  
1951 /ovffiles/setup/resources.zip.

## 1952 D.21 Example of BootDeviceSection

1953 In the example below, the Pre-Install configuration specifies the boot source as a  
1954 specific device (network), while the Post-Install configuration specifies a device  
1955 type (hard disk).

1956 EXAMPLE:

```

1957 <Envelope>
1958 ...
1959 <VirtualSystem ovf:id="...">
1960 ...
1961 <ovf:BootDeviceSection>
1962 <Info>Boot device order specification</Info>
1963 <bootc:CIM_BootConfigSetting>
1964 <bootc:Caption>Pre-Install</bootc:Caption>
1965 <bootc:Description>Boot Sequence for fixup of disk</bootc:Description>
1966 <boots:CIM_BootSourceSetting>
1967 <boots:Caption>Fix-up DVD on the network</boots:Caption>
1968 <boots:InstanceID>3</boots:InstanceID> <!-- Network device-->
1969 </boots:CIM_BootSourceSetting>
1970 <boots:CIM_BootSourceSetting>
1971 <boots:Caption>Boot virtual disk</boots:Caption>
1972 <boots:StructuredBootString>CIM:Hard-Disk</boots:StructuredBootString>
1973 </boots:CIM_BootSourceSetting>
1974 </bootc:CIM_BootConfigSetting>
1975 </ovf:BootDeviceSection>

```

```

1976 ...
1977 </VirtualSystem>
1978 </Envelope>

```

## 1979 D.22 Example of SharedDiskSection

```

1980 EXAMPLE:
1981 <ovf:SharedDiskSection>
1982 <Info>Describes the set of virtual disks shared between VSs</Info>
1983 <ovf:SharedDisk ovf:diskId="datadisk" ovf:fileRef="data"
1984 ovf:capacity="8589934592" ovf:populatedSize="3549324972"
1985 ovf:format=
1986 "http://www.vmware.com/interfaces/specifications/vmdk.html#sparse"/>
1987 <ovf:SharedDisk ovf:diskId="transientdisk" ovf:capacity="536870912"/>
1988 </ovf:SharedDiskSection>

```

## 1989 D.23 Example of ScaleOutSection

```

1990 EXAMPLE:
1991 <VirtualSystemCollection ovf:id="web-tier">
1992 ...
1993 <ovf:ScaleOutSection ovf:id="web-server">
1994 <Info>Web tier</Info>
1995 <ovf:Description>Number of web server instances in web tier</ovf:Description>
1996 <ovf:InstanceCount ovf:default="4" ovf:minimum="2" ovf:maximum="8"/>
1997 </ovf:ScaleOutSection>
1998 ...
1999 <VirtualSystem ovf:id="web-server">
2000 <Info>Prototype web server</Info>
2001 ...
2002 </VirtualSystem>
2003 </VirtualSystemCollection>
2004

```

2005 In the example above, the deployment platform creates a web tier containing between  
2006 two and eight web server virtual system instances, with a default instance count of  
2007 four. The deployment platform makes an appropriate choice (e.g., by prompting the  
2008 user). Assuming three replicas were created, the OVF environment available to the  
2009 guest software in the first replica has the following content structure:

```

2010 EXAMPLE:
2011 <Environment ... ovfenv:id="web-server-1">
2012 ...
2013 <Entity ovfenv:id="web-server-2">
2014 ...
2015 </Entity>
2016 <Entity ovfenv:id="web-server-3">
2017 ...
2018 </Entity>
2019 </Environment>
2020

```

```

2021 EXAMPLE:
2022 <VirtualSystemCollection ovf:id="web-tier">
2023 ...
2024 <DeploymentOptionSection>
2025 <Info>Deployment size options</Info>
2026 <Configuration ovf:id="minimal">
2027

```

```

2028 <Label>Minimal</Label>
2029 <Description>Minimal deployment scenario</Description>
2030 </Configuration>
2031 <Configuration ovf:id="common" ovf:default="true">
2032 <Label>Typical</Label>
2033 <Description>Common deployment scenario</Description>
2034 </Configuration>
2035 ...
2036 </DeploymentOptionSection>
2037 ...
2038 <ovf:ScaleOutSection ovf:id="web-server">
2039 <Info>Web tier</Info>
2040 <ovf:Description>Number of web server instances in web tier</ovf:Description>
2041 <ovf:InstanceCount ovf:default="4"/>
2042 <ovf:InstanceCount ovf:default="1" ovf:configuration="minimal"/>
2043 </ovf:ScaleOutSection>
2044 ...
2045 </VirtualSystemCollection>

```

2046 In the example above, the default replica count is four, unless the minimal  
 2047 deployment scenario is chosen, in which case the default is one.

## 2048 D.24 Example of PlacementGroupSection

2049 EXAMPLE:

```

2050 <Envelope ...>
2051 ...
2052 <ovf:PlacementGroupSection ovf:id="web" ovf:policy="availability">
2053 <Info>Placement policy for group of VSs</Info>
2054 <ovf:Description>Placement policy for web tier</ovf:Description>
2055 </ovf:PlacementGroupSection>
2056 ...
2057 <VirtualSystemCollection ovf:id="web-tier">
2058 ...
2059 <ovf:ScaleOutSection ovf:id="web-node">
2060 <Info>Web tier</Info>
2061 ...
2062 </ovf:ScaleOutSection>
2063 ...
2064 <VirtualSystem ovf:id="web-node">
2065 <Info>Web server</Info>
2066 ...
2067 <ovf:PlacementSection ovf:group="web">
2068 <Info>Placement policy group reference</Info>
2069 </ovf:PlacementSection>
2070 ...
2071 </VirtualSystem>
2072 </VirtualSystemCollection>
2073 </Envelope>

```

2074 In the example above, all virtual systems in the compute tier should be placed  
 2075 separately for high availability. This example also use the ScaleOutSection defined in  
 2076 clause 9.14, in which case each replica get the policy assigned.



## 2077 D.25 Example of EncryptionSection

2078 Below is an example of an OVF encryption section with encryption methods utilized in  
 2079 the OVF document, and the corresponding reference list pointing to the items that have  
 2080 been encrypted.

```

2081
2082 EXAMPLE:
2083 <ovf:EncryptionSection>
2084 <!-- This section contains two different methods of encryption and the corresponding
2085 backpointers to the data that is encrypted ->
2086 <!-- Method#1: Pass phrase based Key derivation ->
2087 <!-- The following derived key block defines PBKDF2 and the corresponding back
2088 pointers to the encrypted data elements -->
2089 <!-- Use a salt value "ovfpassword" and iteration count of 4096 --->
2090 <xenc11:DerivedKey>
2091 <xenc11:KeyDerivationMethod
2092 Algorithm="http://www.rsasecurity.com/rsalabs/pkcs/schemas/pkcs-5#pbkdf2"/>
2093 <pkcs-5:PBKDF2-params>
2094 <Salt>
2095 <Specified>ovfpassword</Specified>
2096 </Salt>
2097 <IterationCount>4096</IterationCount>
2098 <KeyLength>16</KeyLength>
2099 <PRF Algorithm="http://www.w3.org/2001/04/xmldsig-more#hmac-
2100 sha256"/>
2101 </pkcs-5:PBKDF2-params>
2102 ...
2103 <!-- The ReferenceList element below contains references to the file Ref-109.vhd via
2104 the URI syntax which is specified by XML Encryption.
2105 --->
2106 <xenc:ReferenceList>
2107 <xenc:DataReference URI="#first.vhd" />
2108 <xenc:DataReference URI=... />
2109 <xenc:DataReference URI=... />
2110 </xenc:ReferenceList>
2111 </xenc11:DerivedKey>
2112 <!-- Method#2: The following example illustrates use of a symmetric key
2113 transported using the public key within a certificate ->
2114 <xenc:EncryptedKey>
2115 <xenc:EncryptionMethod
2116 Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
2117 <ds:KeyInfo xmlns:ds='http://www.w3.org/2000/09/xmldsig#'
2118 <ds:X509Data>
2119 <ds:X509Certificate> ... </ds:X509Certificate>
2120 </ds:X509Data>
2121 </ds:KeyInfo>
2122 <xenc:CipherData>
2123 <xenc:CipherValue> ... </xenc:CipherValue>
2124 </xenc:CipherData>
2125 <!-- The ReferenceList element below contains reference #second-xml-fragment" to the
2126 XML fragment that has been encrypted using the above method --->
2127 <xenc:ReferenceList>
2128 <xenc:DataReference URI='#second-xml-fragment' />
2129 <xenc:DataReference URI='...' />
2130 <xenc:DataReference URI='...' />
2131 </xenc:ReferenceList>

```

```

2132 </xenc:EncryptedKey>
2133 </ovf:EncryptionSection>
2134 Below is an example of the encrypted file which is referenced in the EncryptionSection
2135 above using URI='Ref-109.vhd' syntax.
2136 EXAMPLE:
2137 <ovf:References>
2138 <ovf:File ovf:id="Xen:9cb10691-4012-4aeb-970c-3d47a906bfff/0b13bdba-3761-8622-22fc-
2139 2e252ed9ce14" ovf:href="Ref-109.vhd">
2140 <!-- the encrypted file referenced by the package is enclosed by an EncryptedData with
2141 a CipherReference to the actual encrypted file. The EncryptionSection in this example
2142 has a back pointer to it under the PBKDF2 algorithm via Id="first.vhd". This tells the
2143 decrypter how to decrypt the file -->
2144 <xenc:EncryptedData Id="first.vhd" Type='http://www.w3.org/2001/04/xmlenc#Element' >
2145 <xenc:EncryptionMethod
2146 Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc" />
2147 <xenc:CipherData>
2148 <xenc:CipherReference URI='Ref-109.vhd' />
2149 </xenc:CipherData>
2150 </xenc:EncryptedData>
2151 </ovf:File>
2152 </ovf:References>
2153 Below is an example of the encrypted OVF markup which is referenced in the
2154 EncryptionSection above using URI='#second-xml-fragment' syntax.
2155 EXAMPLE:
2156 <!-- the EncryptedData element below encompasses encrypted xml from the original
2157 document. It is provided with the Id "first-xml-fragment" which allows it to be
2158 referenced from the EncryptionSection. -->
2159 <xenc:EncryptedData Type=http://www.w3.org/2001/04/xmlenc#Element Id="second-xml-
2160 fragment">
2161 <!-- Each EncryptedData specifies its own encryption method. -->
2162 <xenc:EncryptionMethod Algorithm=http://www.w3.org/2001/04-xmlenc#aes128-cbc/>
2163 <xenc:CipherData>
2164 <!-- Encrypted content --->
2165 <xenc:CipherValue>DEADBEEF</xenc:CipherValue>
2166 </xenc:CipherData>
2167 </xenc:EncryptedData>

```

## 2168 D.26 Example of Internationalization

```

2169 EXAMPLE 1:
2170 <Info ovf:msgid="info.text">Default info.text value if no locale is set or no locale
2171 match</Info>
2172 <License ovf:msgid="license.tomcat-6_0"/> <!-- No default message -->
2173
2174 Using Internal Resource Bundles
2175
2176 EXAMPLE 2:
2177 <ovf:Envelope xml:lang="en-US">
2178 ...
2179 ... sections and content here ...
2180 ...
2181 <Info msgid="info.os">Operating System</Info>
2182 ...
2183 <Strings xml:lang="da-DA">
2184 <Msg ovf:msgid="info.os">Operativsystem</Msg>
2185 ...

```

```

2186 </Strings>
2187 <Strings xml:lang="de-DE">
2188 <Msg ovf:msgid="info.os">Betriebssystem</Msg>
2189 ...
2190 </Strings>
2191 </ovf:Envelope>

```

## 10.2 External Resource Bundles

### EXAMPLE 3:

```

2195 <ovf:Envelope xml:lang="en-US">
2196 <References>
2197 ...
2198 <File ovf:id="it-it-resources" ovf:href="resources/it-it-bundle.msg"/>
2199 </References>
2200 ... sections and content here ...
2201 ...
2202 <Strings xml:lang="it-IT" ovf:fileRef="it-it-resources"/>
2203 ...
2204 </ovf:Envelope>

```

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

```

2207 <Strings
2208 xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1"
2209 xmlns="http://schemas.dmtf.org/ovf/envelope/1"
2210 xml:lang="it-IT">
2211 <Msg ovf:msgid="info.os">Sistema operativo</Msg>
2212 ...
2213 </Strings>

```

## D.27 Example of Message Content in an External File

### EXAMPLE:

```

2216 <Envelope xml:lang="en-US">
2217 <References>
2218 <File ovf:id="license-en-US" ovf:href="license-en-US.txt"/>
2219 <File ovf:id="license-de-DE" ovf:href="license-de-DE.txt"/>
2220 </References>
2221 ...
2222 <VirtualSystem ovf:id="...">
2223 <EulaSection>
2224 <Info>Licensing agreement</Info>
2225 <License ovf:msgid="license">Unused</License>
2226 </EulaSection>
2227 ...
2228 </VirtualSystem>
2229 ...
2230 <Strings xml:lang="en-US">
2231 <Msg ovf:msgid="license" ovf:fileRef="license-en-US">Invalid license</Msg>
2232 </Strings>
2233 <Strings xml:lang="de-DE">
2234 <Msg ovf:msgid="license" ovf:fileRef="license-de-DE">Ihre Lizenz ist nicht
2235 gültig</Msg>
2236 </Strings>
2237 </Envelope>

```

2238 In the example above, the default license agreement is stored in plain text file  
2239 license-en-US.txt, while the license agreement for the de-DE locale is stored in file  
2240 license-de-DE.txt.  
2241 Note that the above mechanism works for all localizable elements and not just License.

## 2242 **D.28 Example of Environment Document**

2243 EXAMPLE: An example of the structure of the OVF environment document follows:  
2244 <?xml version="1.0" encoding="UTF-8"?>  
2245 <Environment xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
2246           xmlns:ovfenv="http://schemas.dmtf.org/ovf/environment/1"  
2247           xmlns="http://schemas.dmtf.org/ovf/environment/1"  
2248           ovfenv:id="identification of VS from OVF descriptor">  
2249     <!-- Information about virtualization platform -->  
2250     <PlatformSection>  
2251         <Kind>Type of virtualization platform</Kind>  
2252         <Version>Version of virtualization platform</Version>  
2253         <Vendor>Vendor of virtualization platform</Vendor>  
2254         <Locale>Language and country code</Locale>  
2255         <TimeZone>Current timezone offset in minutes from UTC</TimeZone>  
2256     </PlatformSection>  
2257     <!-- Properties defined for this virtual system -->  
2258     <PropertySection>  
2259         <Property ovfenv:key="key" ovfenv:value="value">  
2260             <!-- More properties -->  
2261         </PropertySection>  
2262     <Entity ovfenv:id="id of sibling virtual system or virtual system collection">  
2263         <PropertySection>  
2264             <!-- Properties from sibling -->  
2265         </PropertySection>  
2266     </Entity>  
2267 </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 it connects to. 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.

```

2279 <?xml version="1.0" encoding="UTF-8"?>
2280 <Envelope xsi:schemaLocation="http://schemas.dmtf.org/ovf/envelope/2
2281 file:///C:/dsp8023_2.0.0_wgv0.9.5.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2282 xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/2" xmlns="http://schemas.dmtf.org/ovf/envelope/2"
2283 xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
2284 xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
2285 xmlns:epasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2286 schema/2/CIM_EthernetPortAllocationSettingData"
2287 xmlns:sasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_StorageAllocationSettingData">
2288 <!-- References to all external files -->
2289 <References>
2290 <File ovf:id="file1" ovf:href="vmdisk1.vmdk" ovf:size="2000000000"/>
2291 </References>
2292 <!-- Describes meta-information for all virtual disks in the package -->
2293 <DiskSection>
2294 <Info>Describes the set of virtual disks</Info>
2295 <Disk ovf:diskId="vmdisk1" ovf:fileRef="file1" ovf:capacity="4294967296"
2296 ovf:format="http://www.examplecompany.com/interfaces/specifications/vmdk.html#sparse"/>
2297 </DiskSection>
2298 <!-- Describes all networks used in the package -->
2299 <NetworkSection>
2300 <Info>List of logical networks used in the package</Info>
2301 <Network ovf:name="VS Network">
2302 <Description>The network that the VSs connect to</Description>
2303 <NetworkPortProfile>
2304 <!-- Network port profile describing bandwidth reservation. Network port profile
2305 is identified by UUID. -->
2306 <Item>
2307 <epasd:AllocationUnits>bit / second * 10^9</epasd:AllocationUnits>
2308 <epasd:ElementName>Network Port Profile 1</epasd:ElementName>
2309 <epasd:InstanceID>1</epasd:InstanceID>
2310 <epasd:NetworkPortProfileID>aaaaaaaa-bbbb-cccc-dddd-
2311 eeeeeeeeeee</epasd:NetworkPortProfileID>
2312 <epasd:NetworkPortProfileIDType>3</epasd:NetworkPortProfileIDType>
2313 <epasd:Reservation>1</epasd:Reservation>
2314 </Item>
2315 </NetworkPortProfile>
2316 </Network>
2317 </NetworkSection>
2318 <VirtualSystem ovf:id="vm">
2319 <Info>Describes a virtual system</Info>
2320 <Name>Virtual Appliance One</Name>
2321 <ProductSection>
2322 <Info>Describes product information for the appliance</Info>
2323 <Product>The Great Appliance</Product>
2324 <Vendor>Some Great Corporation</Vendor>
2325 <Version>13.00</Version>
2326 <FullVersion>13.00-b5</FullVersion>
2327 <ProductUrl>http://www.somegreatcorporation.com/greatappliance</ProductUrl>
2328 <VendorUrl>http://www.somegreatcorporation.com/</VendorUrl>
2329 <Property ovf:key="admin.email" ovf:type="string">

```

```

2330 <Description>Email address of administrator</Description>
2331 </Property>
2332 <Property ovf:key="app.ip" ovf:type="string" ovf:defaultValue="192.168.0.10">
2333 <Description>The IP address of this appliance</Description>
2334 </Property>
2335 </ProductSection>
2336 <AnnotationSection ovf:required="false">
2337 <Info>A random annotation on this service. It can be ignored</Info>
2338 <Annotation>Contact customer support if you have any problems</Annotation>
2339 </AnnotationSection>
2340 <EulaSection>
2341 <Info>License information for the appliance</Info>
2342 <License>Insert your favorite license here</License>
2343 </EulaSection>
2344 <VirtualHardwareSection>
2345 <Info>Memory = 4 GB, CPU = 1 GHz, Disk = 100 GB, 1 Ethernet nic</Info>
2346 <Item>
2347 <rasd:AllocationUnits>Hertz*10^9</rasd:AllocationUnits>
2348 <rasd:Description>Virtual CPU</rasd:Description>
2349 <rasd:ElementName>1 GHz virtual CPU</rasd:ElementName>
2350 <rasd:InstanceID>1</rasd:InstanceID>
2351 <rasd:Reservation>1</rasd:Reservation>
2352 <rasd:ResourceType>3</rasd:ResourceType>
2353 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2354 </Item>
2355 <Item>
2356 <rasd:AllocationUnits>byte*2^30</rasd:AllocationUnits>
2357 <rasd:Description>Memory</rasd:Description>
2358 <rasd:ElementName>1 GByte of memory</rasd:ElementName>
2359 <rasd:InstanceID>2</rasd:InstanceID>
2360 <rasd:ResourceType>4</rasd:ResourceType>
2361 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2362 </Item>
2363 <EthernetPortItem>
2364 <epasd:Address>00-16-8B-DB-00-5E</epasd:Address>
2365 <epasd:AllocationUnits>bit / second * 10^9 </epasd:AllocationUnits>
2366 <epasd:Connection>VS Network</epasd:Connection>
2367 <epasd:Description>Virtual NIC</epasd:Description>
2368 <epasd:ElementName>Ethernet Port</epasd:ElementName>
2369
2370 <epasd:InstanceID>3</epasd:InstanceID>
2371 <epasd:NetworkPortProfileID>aaaaaaa-bbbb-cccc-dddd-
2372 eeeeeeeee</epasd:NetworkPortProfileID>
2373 <epasd:NetworkPortProfileIDType>3</epasd:NetworkPortProfileIDType>
2374 <epasd:Reservation>1</epasd:Reservation>
2375 <epasd:ResourceType>10</epasd:ResourceType>
2376 <epasd:VirtualQuantityUnits>1</epasd:VirtualQuantityUnits>
2377 </EthernetPortItem>
2378 <StorageItem>
2379 <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
2380 <sasd:Description>Virtual Disk</sasd:Description>
2381 <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
2382 <sasd:InstanceID>4</sasd:InstanceID>
2383 <sasd:Reservation>100</sasd:Reservation>
2384 <sasd:ResourceType>31</sasd:ResourceType>
2385 <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
2386 </StorageItem>
2387 </VirtualHardwareSection>
2388 <OperatingSystemSection ovf:id="58" ovf:required="false">
2389 <Info>Guest Operating System</Info>
2390 <Description>OS</Description>
2391 </OperatingSystemSection>
2392 </VirtualSystem>
2393 </Envelope>

```

## 2394 E.2 Example 2 (OVF Descriptor for One Virtual System and One Network with a 2395 Locally Referenced Network Port Profile)

2396 The example below shows an OVF descriptor that describes a virtual system and a network it connects  
2397 to. The virtual system description in this example uses a network port profile that is described in a local  
2398 file that is contained in the same OVF package. The network described in the network section uses the  
2399 same network port profile description. The network port profile described in this example is used to  
2400 reserve 1 Gbps of bandwidth.

```

2401 <?xml version="1.0" encoding="UTF-8"?>
2402 <Envelope xsi:schemaLocation="http://schemas.dmtf.org/ovf/envelope/2
2403 file:///C:/dsp8023_2.0.0_wgv0.9.5.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2404 xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/2" xmlns="http://schemas.dmtf.org/ovf/envelope/2"
2405 xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
2406 xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
2407 xmlns:epasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2408 schema/2/CIM_EthernetPortAllocationSettingData"
2409 xmlns:sasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_StorageAllocationSettingData">
2410 <!-- References to all external files -->
2411 <References>
2412 <File ovf:id="file1" ovf:href="vmdisk1.vmdk" ovf:size="2000000000"/>
2413 <File ovf:id="networkportprofile1" ovf:href="NetworkPortProfile1.xml"/>
2414 </References>
2415 <!-- Describes meta-information for all virtual disks in the package -->
2416 <DiskSection>
2417 <Info>Describes the set of virtual disks</Info>
2418 <Disk ovf:diskId="vmdisk1" ovf:fileRef="file1" ovf:capacity="4294967296"
2419 ovf:format="http://www.examplecompany.com/interfaces/specifications/vmdk.html#sparse"/>
2420 </DiskSection>
2421 <!-- Describes all networks used in the package -->
2422 <NetworkSection>
2423 <Info>List of logical networks used in the package</Info>
2424 <Network ovf:name="VS Network">
2425 <Description>The network that VSs connect to</Description>
2426 <NetworkPortProfileURI>file:networkportprofile1</NetworkPortProfileURI>
2427 </Network>
2428 </NetworkSection>
2429 <VirtualSystem ovf:id="vm">
2430 <Info>Describes a virtual system</Info>
2431 <Name>Virtual Appliance One</Name>
2432 <ProductSection>
2433 <Info>Describes product information for the appliance</Info>
2434 <Product>The Great Appliance</Product>
2435 <Vendor>Some Great Corporation</Vendor>
2436 <Version>13.00</Version>
2437 <FullVersion>13.00-b5</FullVersion>
2438 <ProductUrl>http://www.somegreatcorporation.com/greatappliance</ProductUrl>
2439 <VendorUrl>http://www.somegreatcorporation.com/</VendorUrl>
2440 <Property ovf:key="admin.email" ovf:type="string">
2441 <Description>Email address of administrator</Description>
2442 </Property>
2443 <Property ovf:key="app.ip" ovf:type="string" ovf:defaultValue="192.168.0.10">
2444 <Description>The IP address of this appliance</Description>
2445 </Property>
2446 </ProductSection>
2447 <AnnotationSection ovf:required="false">
2448 <Info>A random annotation on this service. It can be ignored</Info>
2449 <Annotation>Contact customer support if you have any problems</Annotation>
2450 </AnnotationSection>
2451 <EulaSection>
2452 <Info>License information for the appliance</Info>
2453 <License>Insert your favorite license here</License>
2454 </EulaSection>
2455 <VirtualHardwareSection>
2456 <Info>Memory = 4 GB, CPU = 1 GHz, Disk = 100 GB, 1 Ethernet nic</Info>
2457 <Item>
2458 <rasd:AllocationUnits>Hertz*10^9</rasd:AllocationUnits>
2459 <rasd:Description>Virtual CPU</rasd:Description>
2460 <rasd:ElementName>1 GHz virtual CPU</rasd:ElementName>

```

```

2461 <rasd:InstanceID>1</rasd:InstanceID>
2462 <rasd:Reservation>1</rasd:Reservation>
2463 <rasd:ResourceType>3</rasd:ResourceType>
2464 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2465 </Item>
2466 <Item>
2467 <rasd:AllocationUnits>byte*2^30</rasd:AllocationUnits>
2468 <rasd:Description>Memory</rasd:Description>
2469 <rasd:ElementName>1 GByte of memory</rasd:ElementName>
2470 <rasd:InstanceID>2</rasd:InstanceID>
2471 <rasd:ResourceType>4</rasd:ResourceType>
2472 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2473 </Item>
2474 <EthernetPortItem>
2475 <epasd:Address>00-16-8B-DB-00-5E</epasd:Address>
2476 <epasd:Connection>VS Network</epasd:Connection>
2477 <epasd:Description>Virtual NIC</epasd:Description>
2478 <epasd:ElementName>Ethernet Port</epasd:ElementName>
2479
2480 <epasd:InstanceID>3</epasd:InstanceID>
2481 <epasd:NetworkPortProfileID>file:networkportprofile1</epasd:NetworkPortProfileID>
2482 <epasd:NetworkPortProfileIDType>2</epasd:NetworkPortProfileIDType>
2483 <epasd:ResourceType>10</epasd:ResourceType>
2484 <epasd:VirtualQuantityUnits>1</epasd:VirtualQuantityUnits>
2485 </EthernetPortItem>
2486 <StorageItem>
2487 <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
2488 <sasd:Description>Virtual Disk</sasd:Description>
2489 <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
2490 <sasd:InstanceID>4</sasd:InstanceID>
2491 <sasd:Reservation>100</sasd:Reservation>
2492 <sasd:ResourceType>31</sasd:ResourceType>
2493 <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
2494 </StorageItem>
2495 </VirtualHardwareSection>
2496 <OperatingSystemSection ovf:id="58" ovf:required="false">
2497 <Info>Guest Operating System</Info>
2498 <Description>OS</Description>
2499 </OperatingSystemSection>
2500 </VirtualSystem>
2501 </Envelope>

```

### 2502 **E.3 Example 3 (OVF Descriptor for One Virtual System and One Network with a** 2503 **Network Port Profile referenced by a URI)**

2504 The example below shows an OVF descriptor that describes a virtual system and a network it connects  
2505 to. The virtual system description in this example uses a network port profile that is described by a URI.  
2506 The network described in the network section uses the same network port profile description. The  
2507 network port profile described in this example is used to reserve 1 Gbps of bandwidth.

```

2508 <?xml version="1.0" encoding="UTF-8"?>
2509 <Envelope xsi:schemaLocation="http://schemas.dmtf.org/ovf/envelope/2
2510 file:///C:/dsp8023 2.0.0 wgv0.9.5.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2511 xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/2" xmlns="http://schemas.dmtf.org/ovf/envelope/2"
2512 xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
2513 xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
2514 xmlns:epasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2515 schema/2/CIM_EthernetPortAllocationSettingData"
2516 xmlns:sasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_StorageAllocationSettingData">
2517 <!-- References to all external files -->
2518 <References>
2519 <File ovf:id="file1" ovf:href="vmdisk1.vmdk" ovf:size="2000000000"/>
2520 </References>
2521 <!-- Describes meta-information for all virtual disks in the package -->
2522 <DiskSection>
2523 <Info>Describes the set of virtual disks</Info>
2524 <Disk ovf:diskId="vmdisk1" ovf:fileRef="file1" ovf:capacity="4294967296"
2525 ovf:format="http://www.examplecompany.com/interfaces/specifications/vmdk.html#sparse"/>
2526 </DiskSection>

```



```

2527 <!-- Describes all networks used in the package -->
2528 <NetworkSection>
2529 <Info>List of logical networks used in the package</Info>
2530 <Network ovf:name="VS Network">
2531 <Description>The network that the VSs connect to</Description>
2532
2533 <NetworkPortProfileURI>http://www.dmtf.org/networkportprofiles/networkportprofile1.xml</Netwo
2534 rkPortProfileURI>
2535 </Network>
2536 </NetworkSection>
2537 <VirtualSystem ovf:id="vm">
2538 <Info>Describes a virtual system</Info>
2539 <Name>Virtual Appliance One</Name>
2540 <ProductSection>
2541 <Info>Describes product information for the appliance</Info>
2542 <Product>The Great Appliance</Product>
2543 <Vendor>Some Great Corporation</Vendor>
2544 <Version>13.00</Version>
2545 <FullVersion>13.00-b5</FullVersion>
2546 <ProductUrl>http://www.somegreatcorporation.com/greatappliance</ProductUrl>
2547 <VendorUrl>http://www.somegreatcorporation.com/</VendorUrl>
2548 <Property ovf:key="admin.email" ovf:type="string">
2549 <Description>Email address of administrator</Description>
2550 </Property>
2551 <Property ovf:key="app.ip" ovf:type="string" ovf:defaultValue="192.168.0.10">
2552 <Description>The IP address of this appliance</Description>
2553 </Property>
2554 </ProductSection>
2555 <AnnotationSection ovf:required="false">
2556 <Info>A random annotation on this service. It can be ignored</Info>
2557 <Annotation>Contact customer support if you have any problems</Annotation>
2558 </AnnotationSection>
2559 <EulaSection>
2560 <Info>License information for the appliance</Info>
2561 <License>Insert your favorite license here</License>
2562 </EulaSection>
2563 <VirtualHardwareSection>
2564 <Info>Memory = 4 GB, CPU = 1 GHz, Disk = 100 GB, 1 Ethernet nic</Info>
2565 <Item>
2566 <rasd:AllocationUnits>Hertz*10^9</rasd:AllocationUnits>
2567 <rasd:Description>Virtual CPU</rasd:Description>
2568 <rasd:ElementName>1 GHz virtual CPU</rasd:ElementName>
2569 <rasd:InstanceID>1</rasd:InstanceID>
2570 <rasd:Reservation>1</rasd:Reservation>
2571 <rasd:ResourceType>3</rasd:ResourceType>
2572 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2573 </Item>
2574 <Item>
2575 <rasd:AllocationUnits>byte*2^30</rasd:AllocationUnits>
2576 <rasd:Description>Memory</rasd:Description>
2577 <rasd:ElementName>1 GByte of memory</rasd:ElementName>
2578 <rasd:InstanceID>2</rasd:InstanceID>
2579 <rasd:ResourceType>4</rasd:ResourceType>
2580 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2581 </Item>
2582 <EthernetPortItem>
2583 <epasd:Address>00-16-8B-DB-00-5E</epasd:Address>
2584 <epasd:Connection>VS Network</epasd:Connection>
2585 <epasd:Description>Virtual NIC</epasd:Description>
2586 <epasd:ElementName>Ethernet Port</epasd:ElementName>
2587
2588 <epasd:InstanceID>3</epasd:InstanceID>
2589
2590 <epasd:NetworkPortProfileID>http://www.dmtf.org/networkportprofiles/networkportprofile1.xml</
2591 epasd:NetworkPortProfileID>
2592 <epasd:NetworkPortProfileIDType>2</epasd:NetworkPortProfileIDType>
2593 <epasd:ResourceType>10</epasd:ResourceType>
2594 <epasd:VirtualQuantityUnits>1</epasd:VirtualQuantityUnits>
2595 </EthernetPortItem>
2596 <StorageItem>
2597 <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>

```

```

2598 <sasd:Description>Virtual Disk</sasd:Description>
2599 <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
2600 <sasd:InstanceID>4</sasd:InstanceID>
2601 <sasd:Reservation>100</sasd:Reservation>
2602 <sasd:ResourceType>31</sasd:ResourceType>
2603 <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
2604 </StorageItem>
2605 </VirtualHardwareSection>
2606 <OperatingSystemSection ovf:id="58" ovf:required="false">
2607 <Info>Guest Operating System</Info>
2608 <Description>OS</Description>
2609 </OperatingSystemSection>
2610 </VirtualSystem>
2611 </Envelope>

```

## 2612 E.4 Example 4 (OVF Descriptor for Two Virtual Systems and One Network with 2613 Two Network Port Profiles referenced by URIs)

2614 The example below shows an OVF descriptor that describes two virtual systems and a network they  
2615 connect to. Each virtual system description in this example uses a network port profile that is described  
2616 by a URI. The network described in the network section uses the same two network port profiles. The two  
2617 network port profiles described in this example are used to reserve 1 Gbps of bandwidth and describe  
2618 general network traffic respectively. Annex E.5 and E.6 are examples of these network port profiles.

```

2619 <?xml version="1.0" encoding="UTF-8"?>
2620 <Envelope xsi:schemaLocation="http://schemas.dmtf.org/ovf/envelope/2
2621 file:///C:/dsp8023_2.0.0_wgv0.9.5.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2622 xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/2" xmlns="http://schemas.dmtf.org/ovf/envelope/2"
2623 xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
2624 xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
2625 xmlns:epasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2626 schema/2/CIM_EthernetPortAllocationSettingData"
2627 xmlns:sasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_StorageAllocationSettingData">
2628 <!-- References to all external files -->
2629 <References>
2630 <File ovf:id="file1" ovf:href="vmdisk1.vmdk" ovf:size="2000000000"/>
2631 </References>
2632 <!-- Describes meta-information for all virtual disks in the package -->
2633 <DiskSection>
2634 <Info>Describes the set of virtual disks</Info>
2635 <Disk ovf:diskId="vmdisk1" ovf:fileRef="file1" ovf:capacity="4294967296"
2636 ovf:format="http://www.examplecompany.com/interfaces/specifications/vmdk.html#sparse"/>
2637 </DiskSection>
2638 <!-- Describes all networks used in the package -->
2639 <NetworkSection>
2640 <Info>List of logical networks used in the package</Info>
2641 <Network ovf:name="VS Network">
2642 <Description>The network that the VSs connect to</Description>
2643 <!-- Network port profile for storage traffic -->
2644
2645 <NetworkPortProfileURI>http://www.dmtf.org/networkportprofiles/networkportprofile1.xml</Netwo
2646 rkPortProfileURI>
2647 <!-- Network port profile for networking traffic -->
2648
2649 <NetworkPortProfileURI>http://www.dmtf.org/networkportprofiles/networkportprofile2.xml</Netwo
2650 rkPortProfileURI>
2651 </Network>
2652 </NetworkSection>
2653 <VirtualSystemCollection ovf:id="vscl">
2654 <Info>Collection of 2 VSs</Info>
2655 <VirtualSystem ovf:id="storage server">
2656 <Info>Describes a virtual system</Info>
2657 <Name>Virtual Appliance One</Name>
2658 <ProductSection>
2659 <Info>Describes product information for the appliance</Info>
2660 <Product>The Great Appliance</Product>
2661 <Vendor>Some Great Corporation</Vendor>
2662 <Version>13.00</Version>
2663 <FullVersion>13.00-b5</FullVersion>

```

```

2664 <ProductUrl>http://www.somegreatcorporation.com/greatappliance</ProductUrl>
2665 <VendorUrl>http://www.somegreatcorporation.com/</VendorUrl>
2666 <Property ovf:key="admin.email" ovf:type="string">
2667 <Description>Email address of administrator</Description>
2668 </Property>
2669 <Property ovf:key="app.ip" ovf:type="string" ovf:defaultValue="192.168.0.10">
2670 <Description>The IP address of this appliance</Description>
2671 </Property>
2672 </ProductSection>
2673 <AnnotationSection ovf:required="false">
2674 <Info>A random annotation on this service. It can be ignored</Info>
2675 <Annotation>Contact customer support if you have any problems</Annotation>
2676 </AnnotationSection>
2677 <EulaSection>
2678 <Info>License information for the appliance</Info>
2679 <License>Insert your favorite license here</License>
2680 </EulaSection>
2681 <VirtualHardwareSection>
2682 <Info>Memory = 4 GB, CPU = 1 GHz, Disk = 100 GB, 1 Ethernet nic</Info>
2683 <Item>
2684 <rasd:AllocationUnits>Hertz*10^9</rasd:AllocationUnits>
2685 <rasd:Description>Virtual CPU</rasd:Description>
2686 <rasd:ElementName>1 GHz virtual CPU</rasd:ElementName>
2687 <rasd:InstanceID>1</rasd:InstanceID>
2688 <rasd:Reservation>1</rasd:Reservation>
2689 <rasd:ResourceType>3</rasd:ResourceType>
2690 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2691 </Item>
2692 <Item>
2693 <rasd:AllocationUnits>byte*2^30</rasd:AllocationUnits>
2694 <rasd:Description>Memory</rasd:Description>
2695 <rasd:ElementName>1 GByte of memory</rasd:ElementName>
2696 <rasd:InstanceID>2</rasd:InstanceID>
2697 <rasd:ResourceType>4</rasd:ResourceType>
2698 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2699 </Item>
2700 <EthernetPortItem>
2701 <epasd:Address>00-16-8B-DB-00-5E</epasd:Address>
2702 <epasd:Connection>VS Network</epasd:Connection>
2703 <epasd:Description>Virtual NIC</epasd:Description>
2704
2705 <epasd:ElementName>Ethernet Port</epasd:ElementName>
2706
2707 <epasd:InstanceID>3</epasd:InstanceID>
2708
2709 <epasd:NetworkPortProfileID>http://www.dmtf.org/networkportprofiles/networkportprofile1.xml</
2710 epasd:NetworkPortProfileID>
2711 <epasd:NetworkPortProfileIDType>2</epasd:NetworkPortProfileIDType>
2712 <epasd:ResourceType>10</epasd:ResourceType>
2713 <epasd:VirtualQuantityUnits>1</epasd:VirtualQuantityUnits>
2714 </EthernetPortItem>
2715 <StorageItem>
2716 <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
2717 <sasd:Description>Virtual Disk</sasd:Description>
2718 <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
2719 <sasd:InstanceID>4</sasd:InstanceID>
2720 <sasd:Reservation>100</sasd:Reservation>
2721 <sasd:ResourceType>31</sasd:ResourceType>
2722 <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
2723 </StorageItem>
2724 </VirtualHardwareSection>
2725 <OperatingSystemSection ovf:id="58" ovf:required="false">
2726 <Info>Guest Operating System</Info>
2727 <Description>OS</Description>
2728 </OperatingSystemSection>
2729 </VirtualSystem>
2730 <VirtualSystem ovf:id="web-server">
2731 <Info>Describes a virtual system</Info>
2732 <Name>Virtual Appliance Two</Name>
2733 <ProductSection>
2734 <Info>Describes product information for the appliance</Info>

```

```

2735 <Product>The Great Appliance</Product>
2736 <Vendor>Some Great Corporation</Vendor>
2737 <Version>13.00</Version>
2738 <FullVersion>13.00-b5</FullVersion>
2739 <ProductUrl>http://www.somegreatcorporation.com/greatappliance</ProductUrl>
2740 <VendorUrl>http://www.somegreatcorporation.com/</VendorUrl>
2741 <Property ovf:key="admin.email" ovf:type="string">
2742 <Description>Email address of administrator</Description>
2743 </Property>
2744 <Property ovf:key="app.ip" ovf:type="string" ovf:defaultValue="192.168.0.10">
2745 <Description>The IP address of this appliance</Description>
2746 </Property>
2747 </ProductSection>
2748 <AnnotationSection ovf:required="false">
2749 <Info>A random annotation on this service. It can be ignored</Info>
2750 <Annotation>Contact customer support if you have any problems</Annotation>
2751 </AnnotationSection>
2752 <EulaSection>
2753 <Info>License information for the appliance</Info>
2754 <License>Insert your favorite license here</License>
2755 </EulaSection>
2756 <VirtualHardwareSection>
2757 <Info>Memory = 4 GB, CPU = 1 GHz, Disk = 100 GB, 1 Ethernet nic</Info>
2758 <Item>
2759 <rasd:AllocationUnits>Hertz*10^9</rasd:AllocationUnits>
2760 <rasd:Description>Virtual CPU</rasd:Description>
2761 <rasd:ElementName>1 GHz virtual CPU</rasd:ElementName>
2762 <rasd:InstanceID>1</rasd:InstanceID>
2763 <rasd:Reservation>1</rasd:Reservation>
2764 <rasd:ResourceType>3</rasd:ResourceType>
2765 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2766 </Item>
2767 <Item>
2768 <rasd:AllocationUnits>byte*2^30</rasd:AllocationUnits>
2769 <rasd:Description>Memory</rasd:Description>
2770 <rasd:ElementName>1 GByte of memory</rasd:ElementName>
2771 <rasd:InstanceID>2</rasd:InstanceID>
2772 <rasd:ResourceType>4</rasd:ResourceType>
2773 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2774 </Item>
2775 <EthernetPortItem>
2776 <epasd:Address>00-16-8B-DB-00-5F</epasd:Address>
2777 <epasd:Connection>VS Network</epasd:Connection>
2778 <epasd:Description>Virtual NIC</epasd:Description>
2779
2780 <epasd:ElementName>Ethernet Port</epasd:ElementName>
2781 <!-- Virtual NIC for networking traffic -->
2782 <epasd:InstanceID>3</epasd:InstanceID>
2783
2784 <epasd:NetworkPortProfileID>http://www.dmtf.org/networkportprofiles/networkportprofile2.xml</
2785 epasd:NetworkPortProfileID>
2786 <epasd:NetworkPortProfileIDType>2</epasd:NetworkPortProfileIDType>
2787 <epasd:ResourceType>10</epasd:ResourceType>
2788 <epasd:VirtualQuantityUnits>1</epasd:VirtualQuantityUnits>
2789 </EthernetPortItem>
2790 <StorageItem>
2791 <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
2792 <sasd:Description>Virtual Disk</sasd:Description>
2793 <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
2794 <sasd:InstanceID>4</sasd:InstanceID>
2795 <sasd:Reservation>100</sasd:Reservation>
2796 <sasd:ResourceType>31</sasd:ResourceType>
2797 <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
2798 </StorageItem>
2799 </VirtualHardwareSection>
2800 <OperatingSystemSection ovf:id="58" ovf:required="false">
2801 <Info>Guest Operating System</Info>
2802 <Description>OS</Description>
2803 </OperatingSystemSection>
2804 </VirtualSystem>
2805 </VirtualSystemCollection>

```

2806 </Envelope>

## 2807 E.5 Example 5 (networkportprofile1.xml)

2808

2809 Network Port profile example for bandwidth reservation.

```

2810 <?xml version="1.0" encoding="UTF-8"?>
2811 <NetworkPortProfile xsi:schemaLocation="http://schemas.dmtf.org/ovf/networkportprofile/1
2812 http://schemas.dmtf.org/ovf/networkportprofile/1/dsp8049.xsd"
2813 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2814 xmlns="http://schemas.dmtf.org/ovf/networkportprofile/1"
2815 xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
2816 xmlns:epasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2817 schema/2/CIM_EthernetPortAllocationSettingData">
2818 <Item>
2819 <epasd:AllocationUnits>bit / second * 10^9</epasd:AllocationUnits>
2820 <epasd:ElementName>Network Port Profile 1</epasd:ElementName>
2821 <epasd:InstanceID>1</epasd:InstanceID>
2822 <epasd:NetworkPortProfileID>aaaaaaaa-bbbb-cccc-dddd-
2823 eeeeeeeeeeee</epasd:NetworkPortProfileID>
2824 <epasd:NetworkPortProfileIDType>3</epasd:NetworkPortProfileIDType>
2825 <epasd:Reservation>1</epasd:Reservation>
2826 </Item>
2827 </NetworkPortProfile>

```

## 2828 E.6 Example 6 (networkportprofile2.xml)

2829

2830 Network Port Profile example showing priority setting.

```

2831 <?xml version="1.0" encoding="UTF-8"?>
2832 <NetworkPortProfile xsi:schemaLocation="http://schemas.dmtf.org/ovf/networkportprofile/1
2833 http://schemas.dmtf.org/ovf/networkportprofile/1/dsp8049.xsd"
2834 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2835 xmlns="http://schemas.dmtf.org/ovf/networkportprofile/1"
2836 xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
2837 xmlns:epasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2838 schema/2/CIM_EthernetPortAllocationSettingData">
2839 <Item>
2840 <epasd:AllowedPriorities>0</epasd:AllowedPriorities>
2841 <epasd:AllowedPriorities>1</epasd:AllowedPriorities>
2842 <epasd:DefaultPriority>0</epasd:DefaultPriority>
2843 <epasd:ElementName>Network Port Profile 2</epasd:ElementName>
2844 <epasd:InstanceID>2</epasd:InstanceID>
2845 <epasd:NetworkPortProfileID>aaaaaaaa-bbbb-cccc-dddd-
2846 ffffffff</epasd:NetworkPortProfileID>
2847 <epasd:NetworkPortProfileIDType>3</epasd:NetworkPortProfileIDType>
2848 </Item>
2849 </NetworkPortProfile>

```

## ANNEX F (informative)

2850  
2851  
2852  
2853

### Deployment Considerations

2854 This standard defines an OVF package and the main clauses in this standard deal with this subject  
2855 matter. However, there are deployment considerations necessary to meet the expectations of the OVF  
2856 package author. These are listed below.

#### F.1 OVF Package Structure Deployment Considerations

2857 A deployment function shall verify the ovf package signature and should validate the certificate.

#### F.2 Virtual Hardware Deployment Considerations

2860 If there are multiple virtual hardware sections then the deployment function should select the most  
2861 appropriate one for the target virtualization platform.

2862 If no backing is specified for a device that requires a backing, the deployment function shall make an  
2863 appropriate choice, for example, by prompting the user. More than one backing for a device shall not be  
2864 specified.

2865 The deployment function should select the normal value for a resource allocation but may adjust it within  
2866 the specified range. The virtualization management may further alter the resource allocation within the  
2867 specified range for performance tuning.

#### F.3 Core Metadata Sections Deployment Considerations

2868 The sharing of disk blocks at runtime is optional and virtualization platform specific and shall not be visible  
2869 to the guest software.

2871 A virtualization platform may share storage extents to minimize the amount of space required to support  
2872 the virtual systems. If storage extents are shared by the virtualization platform this sharing is not visible to  
2873 the guest software.

2874 If present the AnnotationSection element may be displayed during deployment of the OVF package.

2875 If present, the EULASection(s) shall be displayed and accepted during deployment of an OVF package.  
2876 If automated deployment is used then the deployment function shall have a methodology to provide  
2877 implicit acceptance.

2878 If virtual disks or other files are included by reference the deployment function shall acquire those files  
2879 prior to the virtual system being launched.

2880 If the specified boot source is a device type, then the deployment function should try all the devices of  
2881 that device type specified.

## ANNEX G (informative)

### Bibliography

- 2882  
2883  
2884  
2885
- 2886 ISO 9660, *Joliet Extensions Specification*, May 1995,  
2887 <http://littlesvr.ca/isomaster/resources/JolietSpecification.html> W3C, *Best Practices for XML*  
2888 *Internationalization*, October 2008,  
2889 <http://www.w3.org/TR/2008/NOTE-xml-i18n-bp-20080213/>
- 2890 DMTF DSP1044, *Processor Device Resource Virtualization Profile 1.0*  
2891 [http://www.dmtf.org/standards/published\\_documents/DSP1044\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP1044_1.0.pdf)
- 2892 DMTF DSP1045, *Memory Resource Virtualization Profile 1.0*  
2893 [http://www.dmtf.org/standards/published\\_documents/DSP1045\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP1045_1.0.pdf)
- 2894 DMTF DSP1047, *Storage Resource Virtualization Profile 1.0*  
2895 [http://www.dmtf.org/standards/published\\_documents/DSP1047\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP1047_1.0.pdf)
- 2896 DMTF DSP1022, *CPU Profile 1.0*,  
2897 [http://www.dmtf.org/standards/published\\_documents/DSP1022\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP1022_1.0.pdf)
- 2898 DMTF DSP1026, *System Memory Profile 1.0*,  
2899 [http://www.dmtf.org/standards/published\\_documents/DSP1026\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP1026_1.0.pdf)
- 2900 DMTF DSP1014, *Ethernet Port Profile 1.0*,  
2901 [http://www.dmtf.org/standards/published\\_documents/DSP1014\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP1014_1.0.pdf)
- 2902 DMTF DSP1050, *Ethernet Port Resource Virtualization Profile 1.1*  
2903 [http://www.dmtf.org/standards/published\\_documents/DSP1050\\_1.1.pdf](http://www.dmtf.org/standards/published_documents/DSP1050_1.1.pdf)
- 2904 DMTF DSP8049, *Network Port Profile XML Schema 1.0*  
2905 [http://schema.dmtf.org/ovf/networkportprofile/1/DSP8049\\_1.0.xsd](http://schema.dmtf.org/ovf/networkportprofile/1/DSP8049_1.0.xsd)  
2906

## ANNEX H (informative)

### Change Log

| Version | Date       | Description                                                                                                           |
|---------|------------|-----------------------------------------------------------------------------------------------------------------------|
| 1.0.0   | 2009-02-22 | DMTF Standard release                                                                                                 |
| 1.1.0   | 2010-01-12 | DMTF Standard release                                                                                                 |
| 2.0.0a  | 2011-06-28 | Work in Progress release                                                                                              |
| 2.0.0b  | 2011-12-14 | Work in Progress release                                                                                              |
| 2.0.0c  | 2012-05-24 | Work in Progress release                                                                                              |
| 2.0.0d  | 2012-09-20 | Work in Progress release                                                                                              |
| 2.0.0   | 2012-10-29 | DMTF Standard release                                                                                                 |
| 2.1.0a  | 2013-05-06 | work in progress release                                                                                              |
| 2.1.0b  | 2013-05-06 | work in progress release                                                                                              |
| 2.1.0   | 2013-05-14 | wgv 0.7.0 clean base line to start 2.1 feature additions                                                              |
| 2.1.0   | 2013-05-14 | wgv 0.7.1 add 2.10 new features<br>added manifest change from 2013-01-10 minutes;<br>added placement policy proposal; |
| 2.1.0   | 2013-06-19 | wgv 0.7.2 added shutdown attribute to 9.4<br>added placement policy update with scoping to 9.15                       |
| 2.1.0   | 2013-06-27 | wgv 0.7.3 added placement policy; removed virtual machine; updated<br>definition for virtual system                   |
| 2.1.0c  | 2013-07-02 | work in progress release                                                                                              |

2907  
2908  
2909  
2910

2911  
2912