



1
2
3
4

Document Number: DSP0243

Date: 2015-08-27

Version: 2.1.1

5 **Open Virtualization Format Specification**

6 **Supersedes: 2.1.0**

7 **Document Class: Normative**

8 **Document Status: Published**

9 **Document Language: en-US**

10 Copyright notice

11 Copyright © 2010-2015 Distributed Management Task Force, Inc. (DMTF). All rights reserved.

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

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26 withdrawn or modified after publication, and shall be indemnified and held harmless by any party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

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

32 This document's normative language is English. Translation into other languages is permitted.

CONTENTS

34	Foreword	6
35	Introduction.....	7
36	1 Scope	9
37	2 Normative references	9
38	3 Terms and definitions	10
39	4 Symbols and abbreviated terms.....	12
40	5 OVF package	13
41	5.1 OVF package structure.....	13
42	5.2 Virtual disk formats	14
43	5.3 OVF package options	14
44	5.4 Distribution as a set of files.....	15
45	6 OVF descriptor	15
46	7 Envelope element.....	16
47	7.1 File references	16
48	7.2 Content element.....	17
49	7.3 Extensibility	18
50	7.4 Conformance	18
51	8 Virtual hardware description.....	19
52	8.1 VirtualHardwareSection	19
53	8.2 Extensibility	20
54	8.3 Virtual hardware elements	21
55	8.4 Ranges on elements.....	22
56	9 Core metadata sections	24
57	9.1 DiskSection	25
58	9.2 NetworkSection.....	26
59	9.3 ResourceAllocationSection	26
60	9.4 AnnotationSection.....	27
61	9.5 ProductSection.....	27
62	9.5.1 Property elements.....	28
63	9.6 EulaSection.....	30
64	9.7 StartupSection	30
65	9.8 DeploymentOptionSection	31
66	9.9 OperatingSystemSection	32
67	9.10 InstallSection.....	32
68	9.11 EnvironmentFilesSection	33
69	9.12 BootDeviceSection.....	33
70	9.13 SharedDiskSection	34
71	9.14 ScaleOutSection	34
72	9.15 PlacementGroupSection and PlacementSection.....	35
73	9.16 EncryptionSection	37
74	10 Internationalization	38
75	10.1 Internal resource bundles	39
76	10.2 External resource bundles	39
77	10.3 Message content in external file	39
78	11 OVF environment and OVF environment file	39
79	11.1 Transport media.....	40
80	11.2 Transport media type	41
81	ANNEX A (informative) Symbols and conventions	42
82	ANNEX B (normative) OVF XSD	43
83	ANNEX C (informative) OVF mime type registration template	44

84	ANNEX D (informative) OVF examples	46
85	D.1 Examples of OVF package structure	46
86	D.2 Examples of distribution of files	46
87	D.3 Example of envelope element	47
88	D.4 Example of file references	48
89	D.5 Example of content element	48
90	D.6 Examples of extensibility	48
91	D.7 Examples of VirtualHardwareSection	49
92	D.8 Examples of virtual hardware elements	50
93	D.9 Example of ranges on elements	50
94	D.10 Example of DiskSection	51
95	D.11 Example of NetworkSection	51
96	D.12 Example of ResourceAllocationSection	52
97	D.13 Example of annotation	52
98	D.14 Example of Product section	52
99	D.15 Example of EULA section	53
100	D.16 Example of StartupSection	53
101	D.17 Example of DeploymentOptionSection	53
102	D.18 Example of OperatingSystemSection	54
103	D.19 Example of InstallSection	54
104	D.20 Example of EnvironmentFilesSection	55
105	D.21 Example of BootDeviceSection	55
106	D.22 Example of SharedDiskSection	56
107	D.23 Example of ScaleOutSection	56
108	D.24 Example of PlacementGroupSection	57
109	D.25 Example of EncryptionSection	58
110	D.26 Example of internationalization	59
111	D.27 Example of message content in an external file	60
112	D.28 Example of environment document	61
113	ANNEX E (informative) Network port profile examples	62
114	E.1 Example 1 (OVF descriptor for one virtual system and one network with an inlined network port profile)	62
115	E.2 Example 2 (OVF descriptor for one virtual system and one network with a locally referenced network port profile)	64
116	E.3 Example 3 (OVF descriptor for one virtual system and one network with a network port profile referenced by a URI)	65
117	E.4 Example 4 (OVF descriptor for two virtual systems and one network with two network port profiles referenced by URIs)	67
118	E.5 Example 5 (networkportprofile1.xml)	70
119	E.6 Example 6 (networkportprofile2.xml)	70
120	ANNEX F (informative) Deployment considerations	71
121	F.1 OVF package structure deployment considerations	71
122	F.2 Virtual hardware deployment considerations	71
123	F.3 Core metadata sections deployment considerations	71
124	ANNEX G (informative) Change log	72
125		
126		
127		
128		
129		

130 **Tables**

131 Table 1 – XML namespace prefixes 16

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

133 Table 3 – HostResource element 21

134 Table 4 – Elements for virtual devices and controllers 22

135 Table 5 – Core metadata sections 24

136 Table 6 – Property types 29

137 Table 7 – Property qualifiers 30

138 Table 8 – Availability attributes 36

139 Table 9 – Affinity Attributes 37

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

141 Table 11 – Core sections for OEF 40

142

143

Foreword

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

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

148

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

187

Introduction

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

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

192 • **Optimized for distribution**

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

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

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

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

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

203 • **Portable VS packaging**

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

208 • **Vendor and platform independent**

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

211 • **Extensible**

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

215 • **Localizable**

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

219 • **Open standard**

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

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

224

225

226

Open Virtualization Format Specification

227 1 Scope

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

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

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

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

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

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

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

244 2 Normative references

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

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

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

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

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

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

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

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

- 263 DMTF DSP1050, Ethernet Port Resource Virtualization Profile 1.0,
264 http://www.dmtf.org/standards/published_documents/DSP1050_1.0.pdf
- 265 DMTF DSP1057, *Virtual System Profile 1.0*,
266 http://www.dmtf.org/standards/published_documents/DSP1057_1.0.pdf
- 267 DMTF DSP8023, *OVF XML Schema Specification for OVF Envelope 2.0*,
268 http://schemas.dmtf.org/ovf/envelope/2/dsp8023_2.0.xsd
- 269 DMTF DSP8027, *OVF XML Schema Specification for OVF Environment 1.0*,
270 http://schemas.dmtf.org/ovf/environment/1/dsp8027_1.0.1.xsd
- 271 DMTF DSP8049, *Network Port Profile XML Schema*,
272 http://schemas.dmtf.org/ovf/networkportprofile/1/dsp8049_1.0.1.xsd
- 273 IETF RFC1738, T. Berners-Lee, *Uniform Resource Locators (URL)*, December 1994,
274 <http://tools.ietf.org/html/rfc1738>
- 275 IETF RFC1952, P. Deutsch, *GZIP file format specification version 4.3*, May 1996,
276 <http://tools.ietf.org/html/rfc1952>
- 277 IETF RFC2616, R. Fielding et al, *Hypertext Transfer Protocol – HTTP/1.1*, June 1999,
278 <http://tools.ietf.org/html/rfc2616>
- 279 IETF Standard 66, *Uniform Resource Identifiers (URI): Generic Syntax*,
280 <http://tools.ietf.org/html/rfc3986>
- 281 IETF Standard 68, *Augmented BNF for Syntax Specifications: ABNF*,
282 <http://tools.ietf.org/html/rfc5234>
- 283 ISO 9660, 1988 Information processing-Volume and file structure of CD-ROM for information interchange,
284 http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=17505
- 285 ISO, ISO/IEC Directives, Part 2, *Rules for the structure and drafting of International Standards*,
286 <http://isotc.iso.org/livelink/livelink.exe?func=ll&objId=4230456&objAction=browse&sort=subtype>
- 287 [ISO/IEC/IEEE 9945:2009](#): Information technology -- Portable Operating System Interface (POSIX®) Base
288 Specifications, Issue 7
289 http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50516
- 290 W3C, *XML Schema Part 1: Structures Second Edition*. 28 October 2004. W3C Recommendation. URL:
291 <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/>
- 292 W3C, *XML Schema Part 2: Datatypes Second Edition*. 28 October 2004. W3C Recommendation. URL:
293 <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/>
- 294 W3C, XML Encryption Syntax and Processing Version 1.1, 13 March 2012, W3C Candidate
295 Recommendation
296 <http://www.w3.org/TR/2012/CR-xmlenc-core1-20120313/>
- 297 FIPS 180-2: Secure Hash Standard (SHS)
298 http://www.nist.gov/manuscript-publication-search.cfm?pub_id=902003#

299 3 Terms and definitions

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

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

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

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

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

315 **3.1**

316 **authoring function**

317 the creation of the OVF package

318 **3.2**

319 **chassis**

320 a placement policy as defined in the class CIM_Chassis

321 **3.3**

322 **conditional**

323 indicates requirements to be followed strictly to conform to the document when the specified conditions
324 are met

325 **3.4**

326 **deployment function**

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

328 **3.5**

329 **geographic**

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

331 **3.6**

332 **guest software**

333 the software that runs inside a virtual system

334 **3.7**

335 **mandatory**

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

338 **3.8**

339 **optional**

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

341 **3.9**

342 **rack**

343 a placement policy as defined in the class CIM_Rack

- 344 **3.10**
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 **3.11**
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 **3.12**
354 **OVF descriptor**
355 an XML file that validates to [DSP8023](#) and provides the information needed to deploy the OVF package
- 356 **3.13**
357 **virtualization platform**
358 the hypervisor on which the virtual systems run
- 359 **3.14**
360 **virtual appliance**
361 a service delivered as a software stack that utilizes one or more virtual systems
- 362 **3.15**
363 **virtual hardware**
364 the processor, memory and I/O resources provided by a virtualization platform that supports a virtual
365 system
- 366 **3.16**
367 **virtual system**
368 as defined in the Virtual System Profile plus the guest software if any
- 369 **3.17**
370 **virtual system collection**
371 a collection of virtual systems
- 372 **3.18**
373 **virtualization management**
374 the software that performs resource allocation and management of virtual systems

375 **4 Symbols and abbreviated terms**

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

- 378 **4.1**
379 **CIM**
380 Common Information Model
- 381 **4.2**
382 **IP**
383 Internet Protocol

384 **4.3**
 385 **OVF**
 386 Open Virtualization Format
 387 **4.4**
 388 **VS**
 389 virtual system
 390 **4.5**
 391 **VSC**
 392 virtual system collection
 393

394 **5 OVF package**

395 **5.1 OVF package structure**

396 An OVF package shall consist of the following files:

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

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

403 An OVF package can be stored as either a single compressed file (`.ova`) or a set of files, as described in
 404 5.3 and 5.4. Both modes shall be supported.

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

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

413 The format of the manifest file is as follows:

```

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

422 See D.1 for an example.

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

426 See ANNEX F for deployment considerations.

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

```

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

```

442 See D.1 for an example.

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

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

450 5.2 Virtual disk formats

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

457 5.3 OVF package options

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

461 All file references in the OVF descriptor are relative-path references and are described in clause 7.1.
 462 Entries in a compressed OVF package shall exist only once.

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

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

468 or

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

475 or

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

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

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

487 **5.4 Distribution as a set of files**

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

489 **6 OVF descriptor**

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

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

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

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

499

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

500 7 Envelope element

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

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

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

513 See D.3 for an example.

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

517 7.1 File references

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

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

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

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

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

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

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

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

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

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

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

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

562 7.2 Content element

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

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

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

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

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

580 7.3 Extensibility

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

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

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

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

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

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

602 See D.6 for an example.

603 7.4 Conformance

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

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

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

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

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

618 **8 Virtual hardware description**

619 **8.1 VirtualHardwareSection**

620 The `VirtualHardwareSection` element can be used to describe the virtual hardware used by the virtual
 621 system.

622 This standard allows incomplete virtual hardware descriptions.

623 The virtualization platform may create additional virtual hardware devices.

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

625

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

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

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

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

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

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

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

- 645 • zero or one `System` elements
- 646 • zero or more `Item` elements
- 647 • zero or more `EthernetPortItem` elements
- 648 • zero or more `StorageItem` elements.

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

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

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

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

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

669 8.2 Extensibility

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

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

676 **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

677 **8.3 Virtual hardware elements**

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

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

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

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

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

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

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

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

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

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

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

713 **Table 3 – HostResource element**

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

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

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

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

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

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

722 8.4 Ranges on elements

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

726 See ANNEX F for virtual hardware deployment considerations.

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

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

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

734 The semantics of range markers are as follows:

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

749 See D.9 for an example.

750

751 **9 Core metadata sections**752 Table 5 shows the core metadata sections that are defined in the `ovf` namespace.753 **Table 5 – Core metadata sections**

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

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

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

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

759 9.1 DiskSection

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

763 The virtual disks in the `DiskSection` element may be referenced by one or more virtual systems.
 764 However, as seen from the guest software, each virtual system gets individual private disks. Any level of
 765 sharing done at runtime is virtualization platform specific and not visible to the guest software. See clause
 766 9.13 for details about how to configure sharing of a virtual disk at runtime with concurrent access. See
 767 D.10 for an example.

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

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

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

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

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

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

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

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

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

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

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

808 9.2 NetworkSection

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

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

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

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

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

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

824 9.3 ResourceAllocationSection

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

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

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

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

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

835 9.4 AnnotationSection

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

838 The `AnnotationSection` element is a valid element for the `VirtualSystem` and the
839 `VirtualSystemCollection` entities.

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

841 9.5 ProductSection

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

844 Zero or more elements may be specified within a `VirtualSystem` element or
845 `VirtualSystemCollection` element.

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

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

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

856 See D.14 for an example.

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

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

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

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

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

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

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

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

867 9.5.1 Property elements

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

927

Table 6 – Property types

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

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

931

Table 7 – Property qualifiers

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

932 9.6 EulaSection

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

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

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

939 9.7 StartupSection

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

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

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

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

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

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

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

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

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

981 9.8 DeploymentOptionSection

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

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

988 The `DeploymentOptionSection` has the following semantics:

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

998 Configurations may be used to control resources for virtual hardware and for virtual system collections.
 999 The `Item`, `EthernetPortItem`, and `StorageItem` elements in `VirtualHardwareSection` elements
 1000 describe resources for `VirtualSystem` entities, while the `Item`, `EthernetPortItem`, and `StorageItem`
 1001 elements in `ResourceAllocationSection` elements describe resources for virtual system collections. For

1002 these two `Item`, `EthernetPortItem`, or `StorageItem` types, the following additional semantics are
1003 defined:

- 1004 • Each `Item`, `EthernetPortItem`, and `StorageItem` has an optional `ovf:configuration`
1005 attribute, containing a list of configurations separated by a single space character. If not
1006 specified, the item shall be selected for any configuration. If specified, the item shall be selected
1007 only if the chosen configuration ID is in the list. A configuration attribute shall not contain a
1008 configuration ID that is not specified in the `DeploymentOptionSection`.
- 1009 • Within a single `VirtualHardwareSection` or `ResourceAllocationSection`, multiple `Item`,
1010 `EthernetPortItem`, and `StorageItem` elements are allowed to refer to the same `InstanceID`. A
1011 single combined `Item`, `EthernetPortItem`, or `StorageItem` for the given `InstanceID` shall be
1012 constructed by picking up the child elements of each `Item`, `EthernetPortItem`, or `StorageItem`
1013 element, with child elements of a former `Item`, `EthernetPortItem`, or `StorageItem` element in
1014 the OVF descriptor not being picked up if there is a like-named child element in a latter `Item`,
1015 `EthernetPortItem`, or `StorageItem` element. Any attributes specified on child elements of
1016 `Item`, `EthernetPortItem`, or `StorageItem` elements that are not picked up that way, are not
1017 part of the combined `Item`, `EthernetPortItem`, or `StorageItem` element.
- 1018 • All `Item`, `EthernetPortItem`, `StorageItem` elements shall specify `ResourceType`, and `Item`,
1019 `EthernetPortItem`, and `StorageItem` elements with the same `InstanceID` shall agree on
1020 `ResourceType`.

1021 Note that the attributes `ovf:configuration` and `ovf:bound` on `Item` may be used in combination to
1022 provide flexible configuration options.

1023 Configurations can further be used to control default values for properties and whether properties are
1024 user configurable. For `Property` elements inside a `ProductSection`, the following additional semantic is
1025 defined:

- 1026 • It is possible to specify alternative default property values for different configurations in a
1027 `DeploymentOptionSection`. In addition to a `Label` and `Description` element, each `Property`
1028 element may optionally contain `Value` elements. The `Value` element shall have an `ovf:value`
1029 attribute specifying the alternative default and an `ovf:configuration` attribute specifying the
1030 configuration in which this new default value should be used. Multiple `Value` elements shall not
1031 refer to the same configuration.
- 1032 • A `Property` element may optionally have an `ovf:configuration` attribute specifying the
1033 configuration in which this property should be user configurable. The value of
1034 `ovf:userConfigurable` is implicitly set to `FALSE` for all other configurations, in which case the
1035 default value of the property may not be changed during installation.

1036 9.9 OperatingSystemSection

1037 An `OperatingSystemSection` specifies the operating system installed on a virtual system. See D.18 for
1038 an example.

1039 The values for `ovf:id` should be taken from the `ValueMap` of the `CIM_OperatingSystem.OsType`
1040 property. The description should be taken from the corresponding `Values` of the
1041 `CIM_OperatingSystem.OsType` property.

1042 The `OperatingSystemSection` element is a valid section for a `VirtualSystem` element only.

1043 9.10 InstallSection

1044 The `InstallSection` element, if specified, indicates that the virtual system needs to be booted once in
1045 order to install and/or configure the guest software. The guest software is expected to access the OVF

- 1046 environment during that boot, and to shut down after having completed the installation and/or
1047 configuration of the software, powering off the guest.
- 1048 If the `InstallSection` is not specified, this indicates that the virtual system does not need to be powered
1049 on to complete installation of guest software. See D.19 for an example.
- 1050 The `InstallSection` element shall be valid only for a `VirtualSystem` element.
- 1051 The `ovf:initialBootStopDelay` attribute specifies a delay in seconds to wait for the virtual system to
1052 power off.
- 1053 If the delay expires and the virtual system has not powered off, the deployment function shall indicate a
1054 failure.
- 1055 An `ovf:initialBootStopDelay` attribute value of zero indicates that the boot stop delay is not specified.
- 1056 Note that the guest software in the virtual system can do multiple reboots before powering off.
- 1057 Several virtual systems in a virtual system collection may have an `InstallSection` element defined, in
1058 which case the above step is done for each virtual system that has an `InstallSection` element.

1059 9.11 EnvironmentFilesSection

- 1060 The `EnvironmentFilesSection` enables the OVF package to specify additional environment file(s) (AEF)
1061 besides the virtual disks. These AEFs enable increased flexibility in image customization outside of virtual
1062 disk capture, allowing an OVF package to provide customized solutions by combining existing virtual
1063 disks without modifying them.
- 1064 The AEF contents are neither generated nor validated by the deployment function.
- 1065 The AEFs are included in the transport media generated by the deployment function.
- 1066 The AEFs are conveyed to the guest software using the indicated transport media type. The AEFs and
1067 OVF environment files are intended to use same transport media and transport media type
- 1068 The `EnvironmentFilesSection` shall contain a `File` element with the attributes `ovf:fileRef` and
1069 `ovf:path` for each AEF provided to the guest software.
- 1070 The `ovf:fileRef` attribute shall specify an existing `File` element in the `References` element. The `File`
1071 element is identified by matching its `ovf:id` attribute value with the `ovf:fileRef` attribute value.
- 1072 The `ovf:path` attribute specifies the relative location in the transport media (see clause 11.1) where the
1073 file should be placed, using the syntax of relative-path references in [RFC3986](#).
- 1074 The referenced `File` element in the `References` element identifies the content using one of the URL
1075 schemes "file", "http", or "https". For the "file" scheme, the content is static and included in the
1076 OVF package. See ANNEX F for deployment considerations
- 1077 For details about transport media type, see clause 11.2.

1078 9.12 BootDeviceSection

- 1079 Individual virtual systems use the default device boot order provided by the virtualization platform's virtual
1080 BIOS. The `BootDeviceSection` allows the OVF package author to specify particular boot configurations
1081 and boot order settings. This enables booting from non-default devices, such as a NIC using PXE, a USB
1082 device, or a secondary disk. Moreover, there could be multiple boot configurations with different boot
1083 orders. For example, a virtual disk may need to be patched before it is bootable and a patch ISO image
1084 could be included in the OVF package.

1085 The Common Information Model (CIM) defines artifacts to deal with boot order use cases prevalent in the
1086 industry for BIOSes found in desktops and servers. The boot configuration is defined by the class
1087 `CIM_BootConfigSetting` that in turn contains one or more `CIM_BootSourceSetting` classes as defined
1088 in the CIM Schema. Each class representing the boot source in turn has either the specific device or a
1089 “device type”, such as disk or CD/DVD, as a boot source.

1090 In the context of this specification, the `InstanceID` property of `CIM_BootSourceSetting` is used for
1091 identifying a specific device as the boot source. The `InstanceID` property of the device as specified in the
1092 `Item` description of the device in the `VirtualHardwareSection` element is used to specify the device as
1093 a boot source. In case the source is desired to be a device type, the `StructuredBootString` field is
1094 used to denote the type of device with values defined by the CIM boot control profile. See ANNEX F for
1095 deployment considerations.

1096 See D.21 for an example.

1097 **9.13 SharedDiskSection**

1098 The existing `DiskSection` element in clause 9.1 describes virtual disks in the OVF package. Virtual disks
1099 in the `DiskSection` element can be referenced by multiple virtual systems, but seen from the guest
1100 software, each virtual system gets individual private disks. Any level of sharing done at runtime is
1101 virtualization platform specific and not visible to the guest software.

1102 Certain applications, such as clustered databases, rely on multiple virtual systems sharing the same
1103 virtual disk at runtime. `SharedDiskSection` allows the OVF package to specify `Disk` elements shared by
1104 more than one virtual system at runtime. These virtual disks may be backed by an external `File`
1105 reference, or may be empty virtual disks without backing. It is recommended that the guest software use
1106 cluster-aware file system technology to be able to handle concurrent access. See D.22 for an example.

1107 The `SharedDiskSection` is a valid section at the outermost envelope level only.

1108 Each virtual disk is represented by a `SharedDisk` element that shall be given an identifier using the
1109 `ovf:diskId` attribute; the identifier shall be unique within the combined content of `DiskSection` and
1110 `SharedDiskSection` element. The `SharedDisk` element has the same structure as the `Disk` element in
1111 the `DiskSection` element, with the addition of an `ovf:readOnly` attribute. The `ovf:readOnly` is optional
1112 and states whether shared disk access is read-write, i.e., `FALSE`, or read-only, i.e., `TRUE`.

1113 Shared virtual disks are referenced from virtual hardware by using the `HostResource` element as
1114 described in clause 8.3.

1115 Support of the `SharedDiskSection` element is optional. The virtualization platform should give an
1116 appropriate error message based on the value of the `ovf:required` attribute on the `SharedDiskSection`
1117 element.

1118 **9.14 ScaleOutSection**

1119 The number of virtual systems or collections of virtual system contained in an OVF package is fixed and
1120 determined by the structure inside the `Envelope` element. The `ScaleOutSection` element allows a
1121 `VirtualSystemCollection` element to contain a set of children that are homogeneous with respect to a
1122 prototypical `VirtualSystem` or `VirtualSystemCollection` element. The `ScaleOutSection` element
1123 shall cause the deployment function to replicate the prototype a number of times, thus allowing the
1124 number of instantiated virtual systems to be configured dynamically at deployment time. See D.23 for an
1125 example.

1126 This mechanism enables scaling of virtual system instances at deployment time. Scaling at runtime is not
1127 within the scope of this specification.

- 1128 The `ScaleOutSection` element is a valid section inside `VirtualSystemCollection` element only.
- 1129 The `ovf:id` attribute on `ScaleOutSection` element identifies the virtual system or collection of virtual
1130 systems prototype to be replicated.
- 1131 For the `InstanceCount` element, the `ovf:minimum` and `ovf:maximum` attribute values shall be non-
1132 negative integers and `ovf:minimum` shall be less than or equal to the value of `ovf:maximum`. The
1133 `ovf:minimum` value may be zero in which case the `VirtualSystemCollection` may contain zero
1134 instances of the prototype. If the `ovf:minimum` attribute is not present, it shall be assumed to have a value
1135 of one. If the `ovf:maximum` attribute is not present, it shall be assumed to have a value of unbounded.
1136 The `ovf:default` attribute is required and shall contain a value within the range defined by `ovf:minimum`
1137 and `ovf:maximum`.
- 1138 Each replicated instance shall be assigned a unique `ovf:id` value within the `VirtualSystemCollection`
1139 element. The unique `ovf:id` value shall be constructed from the prototype `ovf:id` value with a sequence
1140 number appended as follows:
- ```
1141 replica-ovf-id = prototype-ovf-id "-" decimal-number
1142 decimal-number = decimal-digit | (decimal-digit decimal-number)
1143 decimal-digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
```
- 1144 The syntax definitions above use ABNF with the exceptions listed in ANNEX A. The first replica shall  
1145 have sequence number one and following sequence numbers shall be incremented by one for each  
1146 replica. Note that after deployment, no `VirtualSystem` will have the prototype `ovf:id` value itself.
- 1147 If the prototype being replicated has a starting order in the `StartupSection`, all replicas shall share this  
1148 value. It is not possible to specify a particular starting sequence among replicas.
- 1149 Property values for `Property` elements in the prototype are prompted once per replica created. If the  
1150 OVF package author requires a property value to be shared among instances, that `Property` may be  
1151 declared at the containing `VirtualSystemCollection` level and referenced by replicas as described in  
1152 clause 9.5.
- 1153 Configurations from the `DeploymentOptionSection` element may be used to control values for  
1154 `InstanceCount` element. The `InstanceCount` element may have an `ovf:configuration` attribute  
1155 specifying the configuration in which this element should be used. Multiple elements shall not refer to the  
1156 same configuration, and a configuration attribute shall not contain an `ovf:id` value that is not specified in  
1157 the `DeploymentOptionSection`. See D.23 for an example.
- 1158 **9.15 PlacementGroupSection and PlacementSection**
- 1159 Guest software may require the deployment of virtual systems with specific proximity needs. There are  
1160 two use cases:
- 1161 1) the ability to specify that two or more virtual systems should be deployed closely together  
1162 because they rely on fast communication or have a common dependency
  - 1163 2) the ability to specify that two or more virtual systems should be deployed on different platforms  
1164 or locations because of high-availability or disaster recovery considerations
- 1165 The `PlacementGroupSection` element allows an OVF package to define a placement policy for a group  
1166 of `VirtualSystems`. The `PlacementSection` element allows the annotation of the elements with  
1167 membership of a particular placement policy group.
- 1168 Zero or more `PlacementGroupSections` may be defined at the `Envelope` level. The `PlacementSection`  
1169 element may be declared at the `VirtualSystem` or `VirtualSystemCollection` level.

1170 Declaring a `VirtualSystemCollection` a member of a placement policy group applies transitively to all  
 1171 child `VirtualSystem` and child `Virtual System Collections` elements provided that no placement  
 1172 policies are specified for the child `VirtualSystem` or `VirtualSystemCollection`.

1173 If a parent `VirtualSystemCollection` and child `VirtualSystem(s)` and/or  
 1174 `VirtualSystemCollection(s)` both have placement policies, the placement policies of the child  
 1175 `VirtualSystems` and/or child `VirtualSystemCollections` should be applied first. Then placement  
 1176 policy of the parent `VirtualSystemCollection` should be applied.

1177 In the event that there is a conflict in the placement policy, the availability policy should override the  
 1178 affinity policy

1179 The `ovf:id` attribute in `PlacementGroupSection` is used to identify a placement policy. The value of the  
 1180 `ovf:id` attribute shall be unique within the OVF package.

1181 Placement policy group membership is specified using the `ovf:group` attribute in the  
 1182 `PlacementSection`. The value of the `ovf:group` attribute shall match the value of an `ovf:id` attribute in  
 1183 a `PlacementGroupSection`. The value of the `ovf:group` attribute shall be a comma-separated text string  
 1184 of placement policy attributes.

1185 This standard defines the placement policies "affinity" and "availability", specified with the required  
 1186 `ovf:policy` attribute on `PlacementGroupSection`.

1187 The set of attributes used for availability and affinity are defined in Table 8 and Table 9.

1188

**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.           |

1189

1190

**Table 9 – Affinity Attributes**

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

1191 The placement policies that can be declared within a `PlacementGroupSection` are combinations of the  
 1192 availability and affinity attributes defined in Table 8 and Table 9. The placement policy is a single string  
 1193 represented by concatenating the valid placement policy combinations using commas as separators.  
 1194 Allowed combinations of affinity and availability attributes is defined in Table 10.

1195

**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            |

1196 The availability of the parent shall be higher availability order than the availability of the child.

1197 If the placement policy is 'availability' without scoping, no availability order is specified.

1198 See D.24 for an example.

1199 **9.16 EncryptionSection**

1200 It is desirable for licensing and other reasons to have an encryption scheme enabling free exchange of  
 1201 OVF appliances while ensuring that only the intended parties can use them. The "[XML Encryption Syntax  
 1202 and Processing](#)" standard is utilized to encrypt either the files in the reference section or any parts of the  
 1203 XML markup of an OVF document.

1204 The various aspects of OVF encryption are as shown below:

1205 1) block encryption

1206 The OVF package shall utilize block encryption algorithms as specified in the "[XML](#)  
1207 [Encryption Syntax and Processing](#)" standard for this purpose.

1208 2) key derivation

1209 The OVF package may use the appropriate key for this purpose. If the key is derived using  
1210 a passphrase, the author shall use one of the key derivations specified in the "[XML](#)  
1211 [Encryption Syntax and Processing](#)" standard.

1212 3) key transport.

1213 If the encryption key is embedded in the OVF package, the specified key transport  
1214 mechanisms shall be used.

1215 This standard defines a section called the `EncryptionSection` as a focal point for the encryption  
1216 functionality. This section provides a single location for placing the encryption-algorithm-related markup  
1217 and the corresponding reference list to point to the OVF content that has been encrypted.

1218 Note that depending on the parts of the OVF package that has been encrypted, an OVF descriptor may  
1219 not validate against the [DSP8023](#) until decrypted. See D.25 for an example.

## 1220 10 Internationalization

1221 The following elements support localizable messages using the optional `ovf:msgid` attribute:

- 1222 • `Info` element on `Content`
- 1223 • `Name` element on `Content`
- 1224 • `Info` element on `Section`
- 1225 • `Annotation` element on `AnnotationSection`
- 1226 • `License` element on `EulaSection`
- 1227 • `Description` element on `NetworkSection`
- 1228 • `Description` element on `OperatingSystemSection`
- 1229 • `Description`, `Product`, `Vendor`, `Label`, and `Category` elements on `ProductSection`
- 1230 • `Description` and `Label` elements on `Property`
- 1231 • `Description` and `Label` elements on `DeploymentOptionSection`
- 1232 • `ElementName`, `Caption` and `Description` subelements on the `System` element in  
1233 `VirtualHardwareSection`
- 1234 • `ElementName`, `Caption` and `Description` subelements on `Item` elements in  
1235 `VirtualHardwareSection`
- 1236 • `ElementName`, `Caption` and `Description` subelements on `Item` elements in  
1237 `ResourceAllocationSection`

1238 The `ovf:msgid` attribute contains an identifier that refers to a message that may have different values in  
1239 different locales. See D.26 for an example.

1240 The `xml:lang` attribute on the `Envelope` element shall specify the default locale for messages in the  
1241 descriptor. The attribute is optional with a default value of "en-US".

## 1242 10.1 Internal resource bundles

1243 Message resource bundles can be internal or external to the OVF descriptor. Internal resource bundles  
1244 are represented as `Strings` elements at the end of the `Envelope` element. See D.26 for an example.

## 1245 10.2 External resource bundles

1246 External resource bundles shall be listed first in the `References` section and referred to from `Strings`  
1247 elements. An external message bundle follows the same schema as the embedded one. Exactly one  
1248 `Strings` element shall be present in an external message bundle, and that `Strings` element shall not  
1249 have an `ovf:fileRef` attribute specified. See D.26 for an example.

1250 The embedded and external `Strings` elements may be interleaved, but they shall be placed at the end of  
1251 the `Envelope` element. If multiple occurrences of a `msg:id` attribute with a given locale occur, a latter  
1252 value overwrites a former.

## 1253 10.3 Message content in external file

1254 The content of all localizable messages may be stored in an external file using the optional `ovf:fileRef`  
1255 attribute on the `Msg` element. For the `License` element on `EulaSection` in particular, this allows inclusion  
1256 of a standard license text file in unaltered form without any XML header or footer.

1257 The `ovf:fileRef` attribute denotes the message content by identifying an existing `File` element in the  
1258 `References` element; the `File` element is identified by matching its `ovf:id` attribute value with the  
1259 `ovf:fileRef` attribute value. The content of an external file referenced using `ovf:fileRef` shall be  
1260 interpreted as plain text in UTF-8 Unicode.

1261 If the referenced file is not available, the embedded content of the `Msg` element shall be used.

1262 The optional `ovf:fileRef` attribute may appear on `Msg` elements in both internal and external `Strings`  
1263 resource bundles. See D.27 for an example.

## 1264 11 OVF environment and OVF environment file

1265 The OVF environment defines how the guest software and the virtualization platform interact. The OVF  
1266 environment enables the guest software to access information about the virtualization platform, such as  
1267 the user-specified values for the properties defined in the OVF descriptor.

1268 [DSP8027](#) is the XML Schema definition file that contains the elements and attributes defining the format  
1269 and semantics of an XML document that constitutes the OVF environment file (OEF). The OEF shall  
1270 validate against [DSP8027](#).

1271 The OEF is created on a per virtual system basis by the deployment function. The basis of the OEF is the  
1272 OVF descriptor, OVF operational metadata, OVF property values, policy metadata, and other  
1273 user-provided values.

1274 The OEF provides the guest software information about the environment that it is being executed in. The  
1275 way that the OEF is conveyed depends on the transport media type. See D.28 for an example.

1276 The value of the `ovfenv:id` attribute of the `Environment` element shall match the value of the `ovf:id`  
1277 attribute of the `VirtualSystem` entity describing this virtual system.

1278 The `PlatformSection` element contains optional information provided by the deployment function. The  
 1279 `Kind`, `Version`, and `Vendor` elements describe the virtualization platform. The `Locale` and `TimeZone`  
 1280 elements describe the current locale and time zone.

1281 The `PropertySection` element contains `Property` elements with key/value pairs corresponding to all  
 1282 properties specified in the OVF descriptor for the current virtual system, as well as properties specified for  
 1283 the immediate parent `VirtualSystemCollection`, if one exists. The environment presents properties as  
 1284 a single list to make it easy for applications to parse. Furthermore, the single list format supports the  
 1285 override semantics that enables a property on a `VirtualSystem` to override a property defined on a  
 1286 parent `VirtualSystemCollection`. The property that is overridden shall not be in the list. If a property in  
 1287 a virtual system and a property in the parent `VirtualSystemCollection` have identical `ovf:key`,  
 1288 `ovf:class`, and `ovf:instance` attribute values the value of the parent property is overridden by the value  
 1289 of the child property; see 9.5. The value of the parent property may be obtained by adding a child  
 1290 property with a different name referencing the parent property with a macro; see 9.5.

1291 An `Entity` element shall exist for each sibling `VirtualSystem` and `VirtualSystemCollection`, if any  
 1292 are present. The value of the `ovfenv:id` attribute of the `Entity` element shall match the value of the  
 1293 `ovf:id` attribute of the sibling entity. The `Entity` elements contain the property key/value pairs in the  
 1294 sibling's OVF environment documents, so the content of an `Entity` element for a particular sibling shall  
 1295 contain the exact `PropertySection` seen by that sibling. This information can be used, for example, to  
 1296 make configuration information, such as IP addresses, available to `VirtualSystems` that are a part of a  
 1297 multitiered application.

1298 Table 11 shows the core sections that are defined.

1299

**Table 11 – Core sections for OEF**

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

1300 The OEF is extensible by providing new section types. The deployment function should ignore unknown  
 1301 section types and elements specified in OEF.

## 1302 11.1 Transport media

1303 The transport media refers to the format used to convey the information to the guest software. The  
 1304 transport media (e.g., ISO image) is generated by the deployment function.

1305 If the transport media type is 'iso', the generated ISO image shall comply with the [ISO 9660](#) specification  
 1306 with support for Joliet extensions.

1307 The transport media shall contain the OVF environment file and any additional environment file(s) for this  
 1308 particular virtual system. The OEF shall be presented as an XML file named `ovf-env.xml` that is  
 1309 contained in the root directory of the transport media. The guest software is now able to access the  
 1310 information.

1311 For additional environment files, the transport media shall have the root location relative to the `ovf:path`  
 1312 attribute in a directory named "ovfiles" contained in the root directory. This provides an access  
 1313 mechanism for the guest software.



1314 Other custom transport media may support this mechanism. Custom transport medium shall specify how  
1315 to access multiple data sources from a root location. See D.20 for an example. The access mechanism  
1316 for the guest software is not specified.

## 1317 11.2 Transport media type

1318 The transport media type refers to a mechanism to convey transport media over a data link or removable  
1319 storage medium (e.g., CD/DVD-ROM) from deployment functions to guest software.

1320 The `iso` transport media type shall support this mechanism.

1321 This standard defines the “iso” transport type to meet the need for interoperability.

1322 The transport media can be communicated in a number of ways to the guest software. These ways are  
1323 called transport media types. The transport media types are specified in the OVF descriptor by the  
1324 `ovf:transport` attribute of `VirtualHardwareSection`. Several transport media types may be specified,  
1325 separated by a single space character, in which case an implementation is free to use any of them.

1326 To enable interoperability, this specification defines an `iso` transport media type, which all  
1327 implementations that support CD-ROM devices are required to support. The `iso` transport media type  
1328 communicates the environment document by making a dynamically generated ISO image available to the  
1329 guest software.

1330 To support the `iso` transport media type, prior to booting a virtual system, an implementation shall make  
1331 an ISO read-only disk image available as backing for a disconnected CD-ROM. If the `iso` transport media  
1332 type is selected for a `VirtualHardwareSection`, at least one disconnected CD-ROM device shall be  
1333 present in this section.

1334 If the virtual system prior to booting had more than one disconnected CD-ROM, the guest software may  
1335 have to scan connected CD-ROM devices in order to locate the ISO image containing the `ovf-env.xml`  
1336 file.

1337 The transport media containing the OVF environment file shall be made available to the guest software  
1338 on every boot of the virtual system.

1339 Support for the `iso` transport media type is not a requirement for virtual hardware architectures or guest  
1340 software that do not have CD-ROM device support.

1341 To be conformant with this specification, any transport media type other than `iso` shall be given by a URI  
1342 that identifies an unencumbered specification on how to use the transport media type. The specification  
1343 need not be machine readable, but it shall be static and unique so that it may be used as a key by  
1344 software reading an OVF descriptor to uniquely determine the transport media type. The specification  
1345 shall be sufficient for a skilled person to properly interpret the transport media type mechanism for  
1346 implementing the protocols. The URIs should be resolvable.

1347

## ANNEX A (informative)

### Symbols and conventions

1348  
1349  
1350  
1351  
1352 XML examples use the XML namespace prefixes that are defined in Table 1. The XML examples use a  
1353 style to not specify namespace prefixes on child elements. Note that XML rules define that child elements  
1354 specified without a namespace prefix are from the namespace of the parent element, and not from the  
1355 default namespace of the XML document. Throughout the document, whitespace within XML element  
1356 values is used for readability. In practice, a service can accept and strip leading and trailing whitespace  
1357 within element values as if whitespace had not been used.

1358 Syntax definitions in this document use Augmented BNF (ABNF) as defined in IETF [RFC5234](#) with the  
1359 following exceptions:

- 1360 • Rules separated by a bar (|) represent choices, instead of using a forward slash (/) as defined in  
1361 ABNF.
- 1362 • Any characters must be processed case sensitively, instead of case-insensitively as defined in  
1363 ABNF.
- 1364 • Whitespace (i.e., the space character U+0020 and the tab character U+0009) is allowed between  
1365 syntactical elements, instead of assembling elements without whitespace as defined in ABNF.

1366

## ANNEX B (normative)

### OVF XSD

1367  
1368  
1369  
1370

1371 Normative copies of the XML Schemas for this specification may be retrieved by resolving the following  
1372 URLs:

1373  
1374 <http://schemas.dmtf.org/ovf/envelope/2/dsp8023.xsd>  
1375 <http://schemas.dmtf.org/ovf/environment/1/dsp8027.xsd>

1376 Any `xs:documentation` content in XML Schemas for this specification is informative and provided only  
1377 for convenience.

1378 Normative copies of the XML Schemas for the WS-CIM mapping ([DSP0230](#)) of  
1379 `CIM_ResourceAllocationSystemSettingsData`, `CIM_VirtualSystemSettingData`,  
1380 `CIM_EthernetPortAllocationSettingData`, `CIM_StorageAllocationSettingData` and  
1381 `CIM_OperatingSystem`, may be retrieved by resolving the following URLs:

1382  
1383 [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)  
1384 [http://schemas.dmtf.org/wbem/wscim/1/cim-  
1385 schema/2/CIM\\_ResourceAllocationSettingData.xsd](http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData.xsd)  
1386 [http://schemas.dmtf.org/wbem/wscim/1/cim-  
1387 schema/2/CIM\\_EthernetPortAllocationSettingData.xsd](http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_EthernetPortAllocationSettingData.xsd)  
1388 [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)

1389 This specification is based on the following CIM MOFs:

1390 `CIM_VirtualSystemSettingData.mof`  
1391 `CIM_ResourceAllocationSettingData.mof`  
1392 `CIM_EthernetPortAllocationSettingData.mof`  
1393 `CIM_StorageAllocationSettingData.mof`  
1394 `CIM_OperatingSystem.mof`  
1395

## ANNEX C (informative)

### OVF mime type registration template

- 1396  
1397  
1398  
1399
- 1400 Registration Template
- 1401 To: ietf-types@iana.org
- 1402 Subject: Registration of media type Application/OVF
- 1403 Type name: Application
- 1404 Subtype name: OVF
- 1405 Required parameters: none
- 1406 Optional parameters: none
- 1407 Encoding considerations: binary
- 1408 Security considerations:
- 1409 • An OVF package contains active content that is expected to be launched in a virtual system.  
1410 The OVF standard, section 5.1 states: “An OVF package may be signed by signing the manifest  
1411 file. The digest of the manifest file is stored in a certificate file with extension .cert file along with  
1412 the base64-encoded X.509 certificate. The .cert file shall have the same base name as the .ovf  
1413 file and be a sibling of the .ovf file. A consumer of the OVF package shall verify the signature  
1414 and should validate the certificate.”
  - 1415 • An OVF package may contain passwords as part of the configuration information. The OVF  
1416 standard, section 9.5 states: “The optional Boolean attribute `ovf:password` indicates that the  
1417 property value may contain sensitive information. The default value is FALSE. OVF  
1418 implementations prompting for property values are advised to obscure these values when  
1419 `ovf:password` is set to TRUE. This is similar to HTML text input of type password. Note that  
1420 this mechanism affords limited security protection only. Although sensitive values are masked  
1421 from casual observers, default values in the OVF descriptor and assigned values in the OVF  
1422 environment are still passed in clear text.”
- 1423 Interoperability considerations:
- 1424 • OVF has demonstrated interoperability via multiple, interoperating implementations in the  
1425 market.
- 1426 Published specification:
- 1427 • DSP0243\_2.0.0.pdf
- 1428 Applications that use this media type:
- 1429 • Implementations of the DMTF Standard: Cloud Infrastructure Management Interface (CIMI)  
1430 (DSP0263\_1.0.0.pdf)
  - 1431 • Implementations of the SNIA Cloud Data Management Interface (CDMI) – OVF Extension
- 1432 Additional information:
- 1433 • Magic number(s): none

- 1434 • File extension(s): .ova
- 1435 • Macintosh file type code(s): none
- 1436 • Person & email address to contact for further information:
- 1437 • Intended usage: (One of COMMON, LIMITED USE or OBSOLETE.)
- 1438 • Restrictions on usage: (Any restrictions on where the media type can be used go here.)
- 1439 • Author:
- 1440 • Change controller:
- 1441

## ANNEX D (informative)

### OVF examples

1442  
1443  
1444  
1445

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

1446  
1447  
1448  
1449  
1450  
1451  
1452  
1453  
1454  
1455

##### 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
```

1456  
1457  
1458  
1459  
1460

##### EXAMPLE 2:

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

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

1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1470

##### EXAMPLE 3:

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

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

1471  
1472  
1473  
1474  
1475

##### 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:

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

1476  
1477  
1478  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486

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

```
MIIBgjCCASwCAQQwDQYJKoZIhvcNAQEEBQAwwODELMAkGA1UEBhMCQVUxDDAKBgNV
BAgTA1FMRDEbMBkGA1UEAxMSU1NMZWF5L3JzYSB0ZXN0IENBMB4XDTEkMTAwOTIz
MzIwNVowXDTk4MDEwNTIzMTIwNVowYDELMAkGA1UEBhMCQVUxDDAKBgNVBAgTA1FM
RDEZMBcGA1UEChMQTlUyY29tIFB0eS4gTHRkLjELMAkGA1UECjMxMzZAZBGNV
BAMTElNTTGVheSBkZW1vIHN1cnZlcjBcMA0GCSqGSIb3DQEBAQUAA0sAMEgCQQC3
LCXcScWua0PFLkHBLm2VejqpA1F4RQ8q0VjRiPafjx/Z/aWH3ipdMVvuJGa/wFXb
/nDFLDlFwP+oCPwhBtVPAgMBAEEwDQYJKoZIhvcNAQEEBQADQQArNFsihWIjBzb0
DcsU0BvL2bvSwJrPEqFlkDq3F4M6EgutL9axEcANWgbbEdAvNJD1dmEmoWny27Pn
Ims6ZOZB
```

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

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

1487  
1488  
1489  
1490  
1491

##### EXAMPLE 1:

An example of an OVF package as a compressed archive:

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

1492 EXAMPLE 2:  
 1493 An example of an OVF package as a set of files on Web server follows:  
 1494 <http://mywebsite/virtualappliances/package.ovf>  
 1495 <http://mywebsite/virtualappliances/vmdisk1.vmdk>  
 1496 <http://mywebsite/virtualappliances/vmdisk2.vmdk>  
 1497 <http://mywebsite/virtualappliances/resource.iso>  
 1498 <http://mywebsite/virtualappliances/de-DE-resources.xml>

### 1499 D.3 Example of envelope element

1500 An example of the structure of an OVF descriptor with the top-level Envelope element  
 1501 follows:  
 1502 `<?xml version="1.0" encoding="UTF-8"?>`  
 1503 `<Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"`  
 1504 `xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-`  
 1505 `schema/2/CIM_VirtualSystemSettingData"`  
 1506 `xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-`  
 1507 `schema/2/CIM_ResourceAllocationSettingData"`  
 1508 `xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/2"`  
 1509 `xmlns="http://schemas.dmtf.org/ovf/envelope/2"`  
 1510 `xml:lang="en-US">`  
 1511 `<References>`  
 1512 `<File ovf:id="de-DE-resources.xml" ovf:size="15240"`  
 1513 `ovf:href="http://mywebsite/virtualappliances/de-DE-resources.xml"/>`  
 1514 `<File ovf:id="file1" ovf:href="vmdisk1.vmdk" ovf:size="180114671"/>`  
 1515 `<File ovf:id="file2" ovf:href="vmdisk2.vmdk" ovf:size="4882023564"`  
 1516 `ovf:chunkSize="2147483648"/>`  
 1517 `<File ovf:id="file3" ovf:href="resource.iso" ovf:size="212148764"`  
 1518 `ovf:compression="gzip"/>`  
 1519 `<File ovf:id="icon" ovf:href="icon.png" ovf:size="1360"/>`  
 1520 `</References>`  
 1521 `<!-- Describes meta-information about all virtual disks in the package -->`  
 1522 `<DiskSection>`  
 1523 `<Info>Describes the set of virtual disks</Info>`  
 1524 `<!-- Additional section content -->`  
 1525 `</DiskSection>`  
 1526 `<!-- Describes all networks used in the package -->`  
 1527 `<NetworkSection>`  
 1528 `<Info>List of logical networks used in the package</Info>`  
 1529 `<!-- Additional section content -->`  
 1530 `</NetworkSection>`  
 1531 `<SomeSection ovf:required="false">`  
 1532 `<Info>A plain-text description of the content</Info>`  
 1533 `<!-- Additional section content -->`  
 1534 `</SomeSection>`  
 1535 `<!-- Additional sections can follow -->`  
 1536 `<VirtualSystemCollection ovf:id="Some Product">`  
 1537 `<!-- Additional sections including VirtualSystem or VirtualSystemCollection-->`  
 1538 `</VirtualSystemCollection >`  
 1539 `<Strings xml:lang="de-DE">`  
 1540 `<!-- Specification of message resource bundles for de-DE locale -->`  
 1541 `</Strings>`  
 1542 `</Envelope>`

## 1543 D.4 Example of file references

1544 EXAMPLE 1:

1545 The following example shows different types of file references:

```
1546 <File ovf:id="disk1" ovf:href="disk1.vmdk"/>
1547 <File ovf:id="disk2" ovf:href="disk2.vmdk" ovf:size="5368709120"
1548 ovf:chunkSize="2147483648"/>
1549 <File ovf:id="iso1" ovf:href="resources/image1.iso"/>
1550 <File ovf:id="iso2" ovf:href="http://mywebsite/resources/image2.iso"/>
```

1551

1552 EXAMPLE 2:

1553 The following example shows manifest entries corresponding to the file references  
1554 above:

```
1555 SHA1(disk1.vmdk)= 3e19644ec2e806f38951789c76f43e4a0ec7e233
1556 SHA1(disk2.vmdk.000000000)= 4f7158731ff434380bf217da248d47a2478e79d8
1557 SHA1(disk2.vmdk.000000001)= 12849daeeaf43e7a89550384d26bd437bb8defaf
1558 SHA1(disk2.vmdk.000000002)= 4cdd21424bd9eeafa4c42112876217de2ee5556d
1559 SHA1(resources/image1.iso)= 72b37ff3fdd09f2a93f1b8395654649b6d06b5b3
1560 SHA1(http://mywebsite/resources/image2.iso)=
1561 d3c2d179011c970615c5cf10b30957d1c4c968ad
```

## 1562 D.5 Example of content element

1563 An example of a VirtualSystem element structure follows:

```
1564 <VirtualSystem ovf:id="simple-app">
1565 <Info>A virtual system</Info>
1566 <Name>Simple Appliance</Name>
1567 <SomeSection>
1568 <!-- Additional section content -->
1569 </SomeSection>
1570 <!-- Additional sections can follow -->
1571 </VirtualSystem>
```

1572

1573 An example of a VirtualSystemCollection element structure follows:

```
1574 <VirtualSystemCollection ovf:id="multi-tier-app">
1575 <Info>A collection of virtual systems</Info>
1576 <Name>Multi-tiered Appliance</Name>
1577 <SomeSection>
1578 <!-- Additional section content -->
1579 </SomeSection>
1580 <!-- Additional sections can follow -->
1581 <VirtualSystem ovf:id="...">
1582 <!-- Additional sections -->
1583 </VirtualSystem>
1584 <!-- Additional VirtualSystem or VirtualSystemCollection elements can follow-->
1585 </VirtualSystemCollection>
```

## 1586 D.6 Examples of extensibility

1587 EXAMPLE 1:

```
1588 <!-- Optional custom section example -->
1589 <otherns:IncidentTrackingSection ovf:required="false">
1590 <Info>Specifies information useful for incident tracking purposes</Info>
1591 <BuildSystem>Acme Corporation Official Build System</BuildSystem>
1592 <BuildNumber>102876</BuildNumber>
1593 <BuildDate>10-10-2008</BuildDate>
1594 </otherns:IncidentTrackingSection>
```



```

1595
1596 EXAMPLE 2:
1597 <!-- Open content example (extension of existing type) -->
1598 <AnnotationSection>
1599 <Info>Specifies an annotation for this virtual system</Info>
1600 <Annotation>This is an example of how a future element (Author) can still be
1601 parsed by older clients</Annotation>
1602 <!-- AnnotationSection extended with Author element -->
1603 <others:Author ovf:required="false">John Smith</others:Author>
1604 </AnnotationSection>
1605
1606 EXAMPLE 3:
1607 <!-- Optional custom attribute example -->
1608 <Network ovf:name="VS network" others:desiredCapacity="1 Gbit/s">
1609 <Description>The main network for VSs</Description>
1610 </Network>

```

## 1611 D.7 Examples of VirtualHardwareSection

```

1612 EXAMPLE 1:
1613 Example of VirtualHardwareSection:
1614 <VirtualHardwareSection>
1615 <Info>Memory = 4 GB, CPU = 1 GHz, Disk = 100 GB, 1 Ethernet nic</Info>
1616 <Item>
1617 <rasd:AllocationUnits>Hertz*10^9</rasd:AllocationUnits>
1618 <rasd:Description>Virtual CPU</rasd:Description>
1619 <rasd:ElementName>1 GHz virtual CPU</rasd:ElementName>
1620 <rasd:InstanceID>1</rasd:InstanceID>
1621 <rasd:Reservation>1</rasd:Reservation>
1622 <rasd:ResourceType>3</rasd:ResourceType>
1623 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
1624 <rasd:VirtualQuantityUnit>Count</ rasd:VirtualQuantityUnit>
1625 </Item>
1626 <Item>
1627 <rasd:AllocationUnits>byte*2^30</rasd:AllocationUnits>
1628 <rasd:Description>Memory</rasd:Description>
1629 <rasd:ElementName>1 GByte of memory</rasd:ElementName>
1630 <rasd:InstanceID>2</rasd:InstanceID>
1631 <rasd:Limit>4</rasd:Limit>
1632 <rasd:Reservation>4</rasd:Reservation>
1633 <rasd:ResourceType>4</rasd:ResourceType>
1634 </Item>
1635 <EthernetPortItem>
1636 <rasd:AllocationUnits>bit / second *2^30 </rasd:AllocationUnits>
1637 <epasd:Connection>VS Network</epasd:Connection>
1638 <epasd:Description>Virtual NIC</epasd:Description>
1639 <epasd:ElementName>Ethernet Port</epasd:ElementName>
1640 <epasd:NetworkPortProfileID>1</epasd:NetworkPortProfileID>
1641 <epasd:NetworkPortProfileIDType>4</epasd:NetworkPortProfileIDType>
1642 <epasd:ResourceType>10</epasd:ResourceType>
1643 <epasd:VirtualQuantity>1</epasd:VirtualQuantity>
1644 <epasd:VirtualQuantityUnits>Count</epasd:VirtualQuantityUnits>
1645 </EthernetPortItem>
1646 <StorageItem>
1647 <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>

```

```

1648 <sasd:Description>Virtual Disk</sasd:Description>
1649 <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
1650 <sasd:Reservation>100</sasd:Reservation>
1651 <sasd:ResourceType>31</sasd:ResourceType>
1652 <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
1653 <sasd:VirtualQuantityUnit>Count</sasd:VirtualQuantityUnit>
1654 </StorageItem>
1655 </VirtualHardwareSection>
1656
1657 EXAMPLE 2:
1658 <rasd:ResourceSubType>buslogic lsilogic</rasd:ResourceSubType>

```

## 1659 D.8 Examples of virtual hardware elements

1660 EXAMPLE 1:  
 1661 The following example shows a description of memory size:

```

1662 <Item>
1663 <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1664 <rasd:Description>Memory Size</rasd:Description>
1665 <rasd:ElementName>256 MB of memory</rasd:ElementName>
1666 <rasd:InstanceID>2</rasd:InstanceID>
1667 <rasd:ResourceType>4</rasd:ResourceType>
1668 <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
1669 </Item>

```

1670  
 1671 EXAMPLE 2:  
 1672 The following example shows a description of a virtual Ethernet adapter:

```

1673 <EthernetPortItem>
1674 <epasd:Address>00-16-8B-DB-00-5E</epasd:Address>
1675 <epasd:Connection>VS Network</epasd:Connection>
1676 <epasd:Description>Virtual NIC</epasd:Description>
1677
1678 <epasd:ElementName>Ethernet Port 1</epasd:ElementName>
1679 <epasd:InstanceID>3</epasd:InstanceID>
1680 <epasd:NetworkPortProfileID>1</epasd:NetworkPortProfileID>
1681 <epasd:NetworkPortProfileIDType>4</epasd:NetworkPortProfileIDType>
1682 <epasd:VirtualQuantityUnits>1</epasd:VirtualQuantityUnits>
1683 </EthernetPortItem>

```

1684  
 1685 EXAMPLE 3:  
 1686 The following example shows a description of a virtual storage:

```

1687 <StorageItem>
1688 <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
1689 <sasd:Description>Virtual Disk</sasd:Description>
1690 <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
1691 <sasd:InstanceID>4</sasd:InstanceID>
1692 <sasd:Reservation>100</sasd:Reservation>
1693 <sasd:ResourceType>31</sasd:ResourceType>
1694 <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
1695 </StorageItem>

```

## 1696 D.9 Example of ranges on elements

1697 EXAMPLE:  
 1698 The following example shows the use of range markers:  
 1699 <VirtualHardwareSection>

```

1700 <Info>...</Info>
1701 <Item>
1702 <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1703 <rasd:ElementName>512 MB memory size</rasd:ElementName>
1704 <rasd:InstanceID>0</rasd:InstanceID>
1705 <rasd:ResourceType>4</rasd:ResourceType>
1706 <rasd:VirtualQuantity>512</rasd:VirtualQuantity>
1707 </Item>
1708 <Item ovf:bound="min">
1709 <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1710 <rasd:ElementName>384 MB minimum memory size</rasd:ElementName>
1711 <rasd:InstanceID>0</rasd:InstanceID>
1712 <rasd:Reservation>384</rasd:Reservation>
1713 <rasd:ResourceType>4</rasd:ResourceType>
1714 </Item>
1715 <Item ovf:bound="max">
1716 <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1717 <rasd:ElementName>1024 MB maximum memory size</rasd:ElementName>
1718 <rasd:InstanceID>0</rasd:InstanceID>
1719 <rasd:Reservation>1024</rasd:Reservation>
1720 <rasd:ResourceType>4</rasd:ResourceType>
1721 </Item>
1722 </VirtualHardwareSection>

```

## 1723 D.10 Example of DiskSection

```

1724 EXAMPLE: The following example shows a description of virtual disks:
1725 <DiskSection>
1726 <Info>Describes the set of virtual disks</Info>
1727 <Disk ovf:diskId="vmdisk1" ovf:fileRef="file1" ovf:capacity="8589934592"
1728 ovf:populatedSize="3549324972"
1729 ovf:format=
1730 "http://www.vmware.com/interfaces/specifications/vmdk.html#sparse">
1731 </Disk>
1732 <Disk ovf:diskId="vmdisk2" ovf:capacity="536870912">
1733 </Disk>
1734 <Disk ovf:diskId="vmdisk3" ovf:capacity="${disk.size}"
1735 ovf:capacityAllocationUnits="byte * 2^30">
1736 </Disk>
1737 </DiskSection>

```

## 1738 D.11 Example of NetworkSection

```

1739 <NetworkSection>
1740 <Info>List of logical networks used in the package</Info>
1741 <Network ovf:name="VS Network">
1742 <Description>The network that the service will be available on</Description>
1743 <NetworkPortProfile>
1744 <Item>
1745 <epasd:AllocationUnits>GigaBits per Second</epasd:AllocationUnits>
1746 <epasd:ElementName>Network Port Profile 1</epasd:ElementName>
1747 <epasd:InstanceID>1</epasd:InstanceID>
1748 <epasd:NetworkPortProfileID>1</epasd:NetworkPortProfileID>
1749 <epasd:NetworkPortProfileIDType>4</epasd:NetworkPortProfileIDType>
1750 <epasd:Reservation>1</epasd:Reservation>
1751 </Item>

```

```

1752 </NetworkPortProfile>
1753 </Network>
1754 </NetworkSection>

```

## 1755 D.12 Example of ResourceAllocationSection

```

1756 <ResourceAllocationSection>
1757 <Info>Defines reservations for CPU and memory for the collection of VSs</Info>
1758 <Item>
1759 <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1760 <rasd:ElementName>300 MB reservation</rasd:ElementName>
1761 <rasd:InstanceID>0</rasd:InstanceID>
1762 <rasd:Reservation>300</rasd:Reservation>
1763 <rasd:ResourceType>4</rasd:ResourceType>
1764 </Item>
1765 <Item ovf:configuration="..." ovf:bound="...">
1766 <rasd:AllocationUnits>hertz * 10^6</rasd:AllocationUnits>
1767 <rasd:ElementName>500 MHz reservation</rasd:ElementName>
1768 <rasd:InstanceID>0</rasd:InstanceID>
1769 <rasd:Reservation>500</rasd:Reservation>
1770 <rasd:ResourceType>3</rasd:ResourceType>
1771 </Item>
1772 <EthernetPortItem>
1773 <epasd:Address>00-16-8B-DB-00-5E</epasd:Address>
1774 <epasd:Connection>VS Network</epasd:Connection>
1775 <epasd:Description>Virtual NIC</epasd:Description>
1776 <epasd:ElementName>Ethernet Port 1</epasd:ElementName>
1777 <epasd:InstanceID>3</epasd:InstanceID>
1778 <epasd:NetworkPortProfileID>1</epasd:NetworkPortProfileID>
1779 <epasd:NetworkPortProfileIDType>4</epasd:NetworkPortProfileIDType>
1780 <epasd:VirtualQuantityUnits>1</epasd:VirtualQuantityUnits>
1781 </EthernetPortItem>
1782 <StorageItem>
1783 <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
1784 <sasd:Description>Virtual Disk</sasd:Description>
1785 <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
1786 <sasd:InstanceID>4</sasd:InstanceID>
1787 <sasd:Reservation>100</sasd:Reservation>
1788 <sasd:ResourceType>31</sasd:ResourceType>
1789 <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
1790 </StorageItem>
1791 </ResourceAllocationSection>

```

## 1792 D.13 Example of annotation

```

1793 <AnnotationSection>
1794 <Info>An annotation on this service. It can be ignored</Info>
1795 <Annotation>Contact customer support if you have any problems</Annotation>
1796 </AnnotationSection >

```

## 1797 D.14 Example of Product section

```

1798 <ProductSection ovf:class="com.mycrm.myservice" ovf:instance="1">
1799 <Info>Describes product information for the service</Info>
1800 <Product>MyCRM Enterprise</Product>
1801 <Vendor>MyCRM Corporation</Vendor>
1802 <Version>4.5</Version>

```

```

1803 <FullVersion>4.5-b4523</FullVersion>
1804 <ProductUrl>http://www.mycrm.com/enterprise</ProductUrl>
1805 <VendorUrl>http://www.mycrm.com</VendorUrl>
1806 <Icon ovf:height="32" ovf:width="32" ovf:mimeType="image/png" ovf:fileRef="icon">
1807 <Category>Email properties</Category>
1808 <Property ovf:key="adminemail" ovf:type="string" ovf:userConfigurable="true">
1809 <Label>Admin email</Label>
1810 <Description>Email address of administrator</Description>
1811 </Property>
1812 <Category>Admin properties</Category>
1813 <Property ovf:key="app_log" ovf:type="string" ovf:value="low"
1814 ovf:userConfigurable="true">
1815 <Description>Loglevel for the service</Description>
1816 </Property>
1817 <Property ovf:key="app_isSecondary" ovf:value="false" ovf:type="boolean">
1818 <Description>Cluster setup for application server</Description>
1819 </Property>
1820 <Property ovf:key="app_ip" ovf:type="string" ovf:value="${appserver-vm}">
1821 <Description>IP address of the application server VS</Description>
1822 </Property>
1823 </ProductSection>

```

## 1824 D.15 Example of EULA section

```

1825 <EulaSection>
1826 <Info>Licensing agreement</Info>
1827 <License>
1828 Lorem ipsum dolor sit amet, ligula suspendisse nulla pretium, rhoncus tempor placerat
1829 fermentum, enim integer ad vestibulum volutpat. Nisl rhoncus turpis est, vel elit,
1830 congue wisi enim nunc ultricies sit, magna tincidunt. Maecenas aliquam maecenas ligula
1831 nostra, accumsan taciti. Sociis mauris in integer, a dolor netus non dui aliquet,
1832 sagittis felis sodales, dolor sociis mauris, vel eu libero cras. Interdum at. Eget
1833 habitasse elementum est, ipsum purus pede porttitor class, ut adipiscing, aliquet sed
1834 auctor, imperdiet arcu per diam dapibus libero duis. Enim eros in vel, volutpat nec
1835 pellentesque leo, scelerisque.
1836 </License>
1837 </EulaSection>

```

## 1838 D.16 Example of StartupSection

```

1839 <StartupSection>
1840 <Item ovf:id="vm1" ovf:order="0" ovf:startDelay="30" ovf:stopDelay="0"
1841 ovf:startAction="powerOn" ovf:waitingForGuest="true"
1842 ovf:stopAction="powerOff"/>
1843 <Item ovf:id="teamA" ovf:order="0"/>
1844 <Item ovf:id="vm2" ovf:order="1" ovf:startDelay="0" ovf:stopDelay="20"
1845 ovf:startAction="powerOn" ovf:stopAction="guestShutdown"/>
1846 </StartupSection>

```

## 1847 D.17 Example of DeploymentOptionSection

```

1848 <DeploymentOptionSection>
1849 <Configuration ovf:id="minimal">
1850 <Label>Minimal</Label>
1851 <Description>Some description</Description>
1852 </Configuration>
1853 <Configuration ovf:id="normal" ovf:default="true">

```

```

1854 <Label>Typical</Label>
1855 <Description>Some description</Description>
1856 </Configuration>
1857 <!-- Additional configurations -->
1858 </DeploymentOptionSection>
1859
1860 EXAMPLE 1: The following example shows a VirtualHardwareSection:
1861 <VirtualHardwareSection>
1862 <Info>...</Info>
1863 <Item>
1864 <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1865 <rasd:ElementName>512 MB memory size and 256 MB reservation</rasd:ElementName>
1866 <rasd:InstanceID>0</rasd:InstanceID>
1867 <rasd:Reservation>256</rasd:Reservation>
1868 <rasd:ResourceType>4</rasd:ResourceType>
1869 <rasd:VirtualQuantity>512</rasd:VirtualQuantity>
1870 </Item>
1871 ...
1872 <Item ovf:configuration="big">
1873 <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1874 <rasd:ElementName>1024 MB memory size and 512 MB reservation</rasd:ElementName>
1875 <rasd:InstanceID>0</rasd:InstanceID>
1876 <rasd:Reservation>512</rasd:Reservation>
1877 <rasd:ResourceType>4</rasd:ResourceType>
1878 <rasd:VirtualQuantity>1024</rasd:VirtualQuantity>
1879 </Item>
1880 </VirtualHardwareSection>

```

```

1882 EXAMPLE 2: The following shows an example ProductSection:
1883 <ProductSection>
1884 <Property ovf:key="app_adminEmail" ovf:type="string" ovf:userConfigurable="true"
1885 ovf:configuration="standard">
1886 <Label>Admin email</Label>
1887 <Description>Email address of service administrator</Description>
1888 </Property>
1889 <Property ovf:key="app_log" ovf:type="string" ovf:value="low"
1890 ovf:userConfigurable="true">
1891 <Label>Loglevel</Label>
1892 <Description>Loglevel for the service</Description>
1893 <Value ovf:value="none" ovf:configuration="minimal">
1894 </Property>
1895 </ProductSection>

```

1896 In the example above, the app\_adminEmail property is only user configurable in the  
1897 standard configuration, while the default value for the app\_log property is changed  
1898 from low to none in the minimal configuration.

## 1899 D.18 Example of OperatingSystemSection

```

1900 <OperatingSystemSection ovf:id="76">
1901 <Info>Specifies the operating system installed</Info>
1902 <Description>Microsoft Windows Server 2008</Description>
1903 </OperatingSystemSection>

```

## 1904 D.19 Example of InstallSection

```

1905 <InstallSection ovf:initialBootStopDelay="300">

```

```

1906 <Info>Specifies that the virtual system needs to be booted once after having
1907 created the guest software in order to install and/or configure the software
1908 </Info>
1909 </InstallSection>

```

## 1910 D.20 Example of EnvironmentFilesSection

1911 EXAMPLE:

```

1912 <Envelope>
1913 <References>
1914 ...
1915 <File ovf:id="config" ovf:href="config.xml" ovf:size="4332"/>
1916 <File ovf:id="resources" ovf:href="http://mywebsite/resources/resources.zip"/>
1917 </References>
1918 ...
1919 <VirtualSystem ovf:id="...">
1920 ...
1921 <ovf:EnvironmentFilesSection ovf:required="false" ovf:transport="iso">
1922 <Info>Config files to be included in OVF environment</Info>
1923 <ovf:File ovf:fileRef="config" ovf:path="setup/cfg.xml"/>
1924 <ovf:File ovf:fileRef="resources" ovf:path="setup/resources.zip"/>
1925 </ovf:EnvironmentFilesSection>
1926 ...
1927 </VirtualSystem>
1928 ...
1929 </Envelope>

```

1930 In the example above, the file config.xml in the OVF package will be copied to the OVF  
 1931 environment ISO image and be accessible to the guest software in location  
 1932 /ovffiles/setup/cfg.xml, while the file resources.zip will be accessible in location  
 1933 /ovffiles/setup/resources.zip.

## 1934 D.21 Example of BootDeviceSection

1935 In the example below, the Pre-Install configuration specifies the boot source as a  
 1936 specific device (network), while the Post-Install configuration specifies a device  
 1937 type (hard disk).

1938 EXAMPLE:

```

1939 <Envelope>
1940 ...
1941 <VirtualSystem ovf:id="...">
1942 ...
1943 <ovf:BootDeviceSection>
1944 <Info>Boot device order specification</Info>
1945 <bootc:CIM_BootConfigSetting>
1946 <bootc:Caption>Pre-Install</bootc:Caption>
1947 <bootc:Description>Boot Sequence for fixup of disk</bootc:Description>
1948 <boots:CIM_BootSourceSetting>
1949 <boots:Caption>Fix-up DVD on the network</boots:Caption>
1950 <boots:InstanceID>3</boots:InstanceID> <!-- Network device-->
1951 </boots:CIM_BootSourceSetting>
1952 <boots:CIM_BootSourceSetting>
1953 <boots:Caption>Boot virtual disk</boots:Caption>
1954 <boots:StructuredBootString>CIM:Hard-Disk</boots:StructuredBootString>
1955 </boots:CIM_BootSourceSetting>
1956 </bootc:CIM_BootConfigSetting>
1957 </ovf:BootDeviceSection>

```

```

1958 ...
1959 </VirtualSystem>
1960 </Envelope>

```

## 1961 D.22 Example of SharedDiskSection

1962 EXAMPLE:

```

1963 <ovf:SharedDiskSection>
1964 <Info>Describes the set of virtual disks shared between VSs</Info>
1965 <ovf:SharedDisk ovf:diskId="datadisk" ovf:fileRef="data"
1966 ovf:capacity="8589934592" ovf:populatedSize="3549324972"
1967 ovf:format=
1968 "http://www.vmware.com/interfaces/specifications/vmdk.html#sparse"/>
1969 <ovf:SharedDisk ovf:diskId="transientdisk" ovf:capacity="536870912"/>
1970 </ovf:SharedDiskSection>

```

## 1971 D.23 Example of ScaleOutSection

1972 EXAMPLE:

```

1973 <VirtualSystemCollection ovf:id="web-tier">
1974 ...
1975 <ovf:ScaleOutSection ovf:id="web-server">
1976 <Info>Web tier</Info>
1977 <ovf:Description>Number of web server instances in web tier</ovf:Description>
1978 <ovf:InstanceCount ovf:default="4" ovf:minimum="2" ovf:maximum="8"/>
1979 </ovf:ScaleOutSection>
1980 ...
1981 <VirtualSystem ovf:id="web-server">
1982 <Info>Prototype web server</Info>
1983 ...
1984 </VirtualSystem>
1985 </VirtualSystemCollection>

```

1987 In the example above, the deployment platform creates a web tier containing between  
1988 two and eight web server virtual system instances, with a default instance count of  
1989 four. The deployment platform makes an appropriate choice (e.g., by prompting the  
1990 user). Assuming three replicas were created, the OVF environment available to the  
1991 guest software in the first replica has the following content structure:

1992 EXAMPLE:

```

1994 <Environment ... ovfenv:id="web-server-1">
1995 ...
1996 <Entity ovfenv:id="web-server-2">
1997 ...
1998 </Entity>
1999 <Entity ovfenv:id="web-server-3">
2000 ...
2001 </Entity>
2002 </Environment>

```

2004 EXAMPLE:

```

2005 <VirtualSystemCollection ovf:id="web-tier">
2006 ...
2007 <DeploymentOptionSection>
2008 <Info>Deployment size options</Info>
2009 <Configuration ovf:id="minimal">

```



```

2010 <Label>Minimal</Label>
2011 <Description>Minimal deployment scenario</Description>
2012 </Configuration>
2013 <Configuration ovf:id="common" ovf:default="true">
2014 <Label>Typical</Label>
2015 <Description>Common deployment scenario</Description>
2016 </Configuration>
2017 ...
2018 </DeploymentOptionSection>
2019 ...
2020 <ovf:ScaleOutSection ovf:id="web-server">
2021 <Info>Web tier</Info>
2022 <ovf:Description>Number of web server instances in web tier</ovf:Description>
2023 <ovf:InstanceCount ovf:default="4"/>
2024 <ovf:InstanceCount ovf:default="1" ovf:configuration="minimal"/>
2025 </ovf:ScaleOutSection>
2026 ...
2027 </VirtualSystemCollection>

```

2028 In the example above, the default replica count is four, unless the minimal deployment  
 2029 scenario is chosen, in which case the default is one.

## 2030 D.24 Example of PlacementGroupSection

2031 EXAMPLE:

```

2032 <Envelope ...>
2033 ...
2034 <ovf:PlacementGroupSection ovf:id="web" ovf:policy="availability">
2035 <Info>Placement policy for group of VSs</Info>
2036 <ovf:Description>Placement policy for web tier</ovf:Description>
2037 </ovf:PlacementGroupSection>
2038 ...
2039 <VirtualSystemCollection ovf:id="web-tier">
2040 ...
2041 <ovf:ScaleOutSection ovf:id="web-node">
2042 <Info>Web tier</Info>
2043 ...
2044 </ovf:ScaleOutSection>
2045 ...
2046 <VirtualSystem ovf:id="web-node">
2047 <Info>Web server</Info>
2048 ...
2049 <ovf:PlacementSection ovf:group="web">
2050 <Info>Placement policy group reference</Info>
2051 </ovf:PlacementSection>
2052 ...
2053 </VirtualSystem>
2054 </VirtualSystemCollection>
2055 </Envelope>

```

2056 In the example above, all virtual systems in the compute tier should be placed  
 2057 separately for high availability. This example also use the ScaleOutSection defined in  
 2058 clause 9.14, in which case each replica get the policy assigned.

## 2059 D.25 Example of EncryptionSection

2060 Below is an example of an OVF encryption section with encryption methods utilized in  
 2061 the OVF document, and the corresponding reference list pointing to the items that have  
 2062 been encrypted.

2063

2064 EXAMPLE:

2065 &lt;ovf:EncryptionSection&gt;

2066 <!-- This section contains two different methods of encryption and the corresponding  
 2067 back pointers to the data that is encrypted -->

2068 &lt;!-- Method#1: Pass phrase based Key derivation --&gt;

2069 <!-- The following derived key block defines PBKDF2 and the corresponding back  
 2070 pointers to the encrypted data elements -->

2071 &lt;!-- Use a salt value "ovfpassword" and iteration count of 4096 ---&gt;

2072 &lt;xenc11:DerivedKey&gt;

2073 &lt;xenc11:KeyDerivationMethod

2074 Algorithm="http://www.rsasecurity.com/rsalabs/pkcs/schemas/pkcs-5#pbkdf2"/&gt;

2075 &lt;pkcs-5:PBKDF2-params&gt;

2076 &lt;Salt&gt;

2077 &lt;Specified&gt;ovfpassword&lt;/Specified&gt;

2078 &lt;/Salt&gt;

2079 &lt;IterationCount&gt;4096&lt;/IterationCount&gt;

2080 &lt;KeyLength&gt;16&lt;/KeyLength&gt;

2081 &lt;PRF Algorithm="http://www.w3.org/2001/04/xmldsig-more#hmac-

2082 sha256"/&gt;

2083 &lt;/pkcs-5:PBKDF2-params&gt;

2084 ...

2085 <!-- The ReferenceList element below contains references to the file Ref-109.vhd via  
 2086 the URI syntax which is specified by XML Encryption.

2087 ---&gt;

2088 &lt;xenc:ReferenceList&gt;

2089 &lt;xenc:DataReference URI="#first.vhd" /&gt;

2090 &lt;xenc:DataReference URI=... /&gt;

2091 &lt;xenc:DataReference URI=... /&gt;

2092 &lt;/xenc:ReferenceList&gt;

2093 &lt;/xenc11:DerivedKey&gt;

2094 <!-- Method#2: The following example illustrates use of a symmetric key  
 2095 transported using the public key within a certificate -->

2096 &lt;xenc:EncryptedKey&gt;

2097 &lt;xenc:EncryptionMethod

2098 Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1\_5"/&gt;

2099 &lt;ds:KeyInfo xmlns:ds='http://www.w3.org/2000/09/xmldsig#'

2100 &lt;ds:X509Data&gt;

2101 &lt;ds:X509Certificate&gt; ... &lt;/ds:X509Certificate&gt;

2102 &lt;/ds:X509Data&gt;

2103 &lt;/ds:KeyInfo&gt;

2104 &lt;xenc:CipherData&gt;

2105 &lt;xenc:CipherValue&gt; ... &lt;/xenc:CipherValue&gt;

2106 &lt;/xenc:CipherData&gt;

2107 <!-- The ReferenceList element below contains reference #second-xml-fragment" to the  
 2108 XML fragment that has been encrypted using the above method --->

2109 &lt;xenc:ReferenceList&gt;

2110 &lt;xenc:DataReference URI='#second-xml-fragment' /&gt;

2111 &lt;xenc:DataReference URI='...' /&gt;

2112 &lt;xenc:DataReference URI='...' /&gt;

```

2113 </xenc:ReferenceList>
2114 </xenc:EncryptedKey>
2115 </ovf:EncryptionSection>
2116 Below is an example of the encrypted file which is referenced in the EncryptionSection
2117 above using URI='Ref-109.vhd' syntax.
2118 EXAMPLE:
2119 <ovf:References>
2120 <ovf:File ovf:id="Xen:9cb10691-4012-4aeb-970c-3d47a906bfff/0b13bdba-3761-8622-22fc-
2121 2e252ed9ce14" ovf:href="Ref-109.vhd">
2122 <!-- the encrypted file referenced by the package is enclosed by an EncryptedData with
2123 a CipherReference to the actual encrypted file. The EncryptionSection in this example
2124 has a back pointer to it under the PBKDF2 algorithm via Id="first.vhd". This tells the
2125 decrypter how to decrypt the file -->
2126 <xenc:EncryptedData Id="first.vhd" Type='http://www.w3.org/2001/04/xmlenc#Element' >
2127 <xenc:EncryptionMethod
2128 Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc" />
2129 <xenc:CipherData>
2130 <xenc:CipherReference URI='Ref-109.vhd' />
2131 </xenc:CipherData>
2132 </xenc:EncryptedData>
2133 </ovf:File>
2134 </ovf:References>
2135 Below is an example of the encrypted OVF markup which is referenced in the
2136 EncryptionSection above using URI='#second-xml-fragment' syntax.
2137 EXAMPLE:
2138 <!-- the EncryptedData element below encompasses encrypted xml from the original
2139 document. It is provided with the Id "first-xml-fragment" which allows it to be
2140 referenced from the EncryptionSection. -->
2141 <xenc:EncryptedData Type=http://www.w3.org/2001/04/xmlenc#Element Id="second-xml-
2142 fragment">
2143 <!-- Each EncryptedData specifies its own encryption method. -->
2144 <xenc:EncryptionMethod Algorithm=http://www.w3.org/2001/04-xmlenc#aes128-cbc/>
2145 <xenc:CipherData>
2146 <!-- Encrypted content --->
2147 <xenc:CipherValue>DEADBEEF</xenc:CipherValue>
2148 </xenc:CipherData>
2149 </xenc:EncryptedData>

```

## 2150 D.26 Example of internationalization

```

2151 EXAMPLE 1:
2152 <Info ovf:msgid="info.text">Default info.text value if no locale is set or no locale
2153 match</Info>
2154 <License ovf:msgid="license.tomcat-6_0"/> <!-- No default message -->
2155
2156 Using Internal Resource Bundles
2157
2158 EXAMPLE 2:
2159 <ovf:Envelope xml:lang="en-US">
2160 ...
2161 ... sections and content here ...
2162 ...
2163 <Info msgid="info.os">Operating System</Info>
2164 ...
2165 <Strings xml:lang="da-DA">

```

```

2166 <Msg ovf:msgid="info.os">Operating System</Msg>
2167 ...
2168 </Strings>
2169 <Strings xml:lang="de-DE">
2170 <Msg ovf:msgid="info.os">Betriebssystem</Msg>
2171 ...
2172 </Strings>
2173 </ovf:Envelope>
2174

```

## 10.2 External Resource Bundles

### EXAMPLE 3:

```

2177 <ovf:Envelope xml:lang="en-US">
2178 <References>
2179 ...
2180 <File ovf:id="it-it-resources" ovf:href="resources/it-it-bundle.msg"/>
2181 </References>
2182 ... sections and content here ...
2183 ...
2184 <Strings xml:lang="it-IT" ovf:fileRef="it-it-resources"/>
2185 ...
2186 </ovf:Envelope>

```

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

```

2189 <Strings
2190 xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1"
2191 xmlns="http://schemas.dmtf.org/ovf/envelope/1"
2192 xml:lang="it-IT">
2193 <Msg ovf:msgid="info.os">Sistema operativo</Msg>
2194 ...
2195 </Strings>

```

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

### EXAMPLE:

```

2198 <Envelope xml:lang="en-US">
2199 <References>
2200 <File ovf:id="license-en-US" ovf:href="license-en-US.txt"/>
2201 <File ovf:id="license-de-DE" ovf:href="license-de-DE.txt"/>
2202 </References>
2203 ...
2204 <VirtualSystem ovf:id="...">
2205 <EulaSection>
2206 <Info>Licensing agreement</Info>
2207 <License ovf:msgid="license">Unused</License>
2208 </EulaSection>
2209 ...
2210 </VirtualSystem>
2211 ...
2212 <Strings xml:lang="en-US">
2213 <Msg ovf:msgid="license" ovf:fileRef="license-en-US">Invalid license</Msg>
2214 </Strings>
2215 <Strings xml:lang="de-DE">
2216 <Msg ovf:msgid="license" ovf:fileRef="license-de-DE">Ihre Lizenz ist nicht
2217 gültig</Msg>
2218 </Strings>

```

2219 </Envelope>  
 2220 In the example above, the default license agreement is stored in plain text file  
 2221 license-en-US.txt, while the license agreement for the de-DE locale is stored in file  
 2222 license-de-DE.txt.  
 2223 Note that the above mechanism works for all localizable elements and not just License.

## 2224 **D.28 Example of environment document**

2225 EXAMPLE: An example of the structure of the OVF environment document follows:  
 2226 <?xml version="1.0" encoding="UTF-8"?>  
 2227 <Environment xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 2228       xmlns:ovfenv="http://schemas.dmtf.org/ovf/environment/1"  
 2229       xmlns="http://schemas.dmtf.org/ovf/environment/1"  
 2230       ovfenv:id="identification of VS from OVF descriptor">  
 2231     <!-- Information about virtualization platform -->  
 2232     <PlatformSection>  
 2233       <Kind>Type of virtualization platform</Kind>  
 2234       <Version>Version of virtualization platform</Version>  
 2235       <Vendor>Vendor of virtualization platform</Vendor>  
 2236       <Locale>Language and country code</Locale>  
 2237       <TimeZone>Current timezone offset in minutes from UTC</TimeZone>  
 2238     </PlatformSection>  
 2239     <!-- Properties defined for this virtual system -->  
 2240     <PropertySection>  
 2241       <Property ovfenv:key="key" ovfenv:value="value">  
 2242        <!-- More properties -->  
 2243       </PropertySection>  
 2244     <Entity ovfenv:id="id of sibling virtual system or virtual system collection">  
 2245       <PropertySection>  
 2246        <!-- Properties from sibling -->  
 2247       </PropertySection>  
 2248     </Entity>  
 2249 </Environment>

## ANNEX E (informative)

### Network port profile examples

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

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

```

2261 <?xml version="1.0" encoding="UTF-8"?>
2262 <Envelope xsi:schemaLocation="http://schemas.dmtf.org/ovf/envelope/2
2263 file:///C:/dsp8023_2.0.0_wgv0.9.5.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2264 xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/2" xmlns="http://schemas.dmtf.org/ovf/envelope/2"
2265 xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
2266 xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
2267 xmlns:epasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2268 schema/2/CIM_EthernetPortAllocationSettingData"
2269 xmlns:sasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_StorageAllocationSettingData">
2270 <!-- References to all external files -->
2271 <References>
2272 <File ovf:id="file1" ovf:href="vmdisk1.vmdk" ovf:size="2000000000"/>
2273 </References>
2274 <!-- Describes meta-information for all virtual disks in the package -->
2275 <DiskSection>
2276 <Info>Describes the set of virtual disks</Info>
2277 <Disk ovf:diskId="vmdisk1" ovf:fileRef="file1" ovf:capacity="4294967296"
2278 ovf:format="http://www.examplecompany.com/interfaces/specifications/vmdk.html#sparse"/>
2279 </DiskSection>
2280 <!-- Describes all networks used in the package -->
2281 <NetworkSection>
2282 <Info>List of logical networks used in the package</Info>
2283 <Network ovf:name="VS Network">
2284 <Description>The network that the VSs connect to</Description>
2285 <NetworkPortProfile>
2286 <!-- Network port profile describing bandwidth reservation. Network port profile
2287 is identified by UUID. -->
2288 <Item>
2289 <epasd:AllocationUnits>bit / second * 10^9</epasd:AllocationUnits>
2290 <epasd:ElementName>Network Port Profile 1</epasd:ElementName>
2291 <epasd:InstanceID>1</epasd:InstanceID>
2292 <epasd:NetworkPortProfileID>aaaaaaaa-bbbb-cccc-dddd-
2293 eeeeeeeeeee</epasd:NetworkPortProfileID>
2294 <epasd:NetworkPortProfileIDType>3</epasd:NetworkPortProfileIDType>
2295 <epasd:Reservation>1</epasd:Reservation>
2296 </Item>
2297 </NetworkPortProfile>
2298 </Network>
2299 </NetworkSection>
2300 <VirtualSystem ovf:id="vm">
2301 <Info>Describes a virtual system</Info>
2302 <Name>Virtual Appliance One</Name>
2303 <ProductSection>
2304 <Info>Describes product information for the appliance</Info>
2305 <Product>The Great Appliance</Product>
2306 <Vendor>Some Great Corporation</Vendor>
2307 <Version>13.00</Version>
2308 <FullVersion>13.00-b5</FullVersion>
2309 <ProductUrl>http://www.somegreatcorporation.com/greatappliance</ProductUrl>
2310 <VendorUrl>http://www.somegreatcorporation.com/</VendorUrl>

```

```

2311 <Property ovf:key="adminemail" ovf:type="string">
2312 <Description>Email address of administrator</Description>
2313 </Property>
2314 <Property ovf:key="app_ip" ovf:type="string" ovf:defaultValue="192.168.0.10">
2315 <Description>The IP address of this appliance</Description>
2316 </Property>
2317 </ProductSection>
2318 <AnnotationSection ovf:required="false">
2319 <Info>A random annotation on this service. It can be ignored</Info>
2320 <Annotation>Contact customer support if you have any problems</Annotation>
2321 </AnnotationSection>
2322 <EulaSection>
2323 <Info>License information for the appliance</Info>
2324 <License>Insert your favorite license here</License>
2325 </EulaSection>
2326 <VirtualHardwareSection>
2327 <Info>Memory = 4 GB, CPU = 1 GHz, Disk = 100 GB, 1 Ethernet nic</Info>
2328 <Item>
2329 <rasd:AllocationUnits>Hertz*10^9</rasd:AllocationUnits>
2330 <rasd:Description>Virtual CPU</rasd:Description>
2331 <rasd:ElementName>1 GHz virtual CPU</rasd:ElementName>
2332 <rasd:InstanceID>1</rasd:InstanceID>
2333 <rasd:Reservation>1</rasd:Reservation>
2334 <rasd:ResourceType>3</rasd:ResourceType>
2335 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2336 </Item>
2337 <Item>
2338 <rasd:AllocationUnits>byte*2^30</rasd:AllocationUnits>
2339 <rasd:Description>Memory</rasd:Description>
2340 <rasd:ElementName>1 GByte of memory</rasd:ElementName>
2341 <rasd:InstanceID>2</rasd:InstanceID>
2342 <rasd:ResourceType>4</rasd:ResourceType>
2343 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2344 </Item>
2345 <EthernetPortItem>
2346 <epasd:Address>00-16-8B-DB-00-5E</epasd:Address>
2347 <epasd:AllocationUnits>bit / second * 10^9 </epasd:AllocationUnits>
2348 <epasd:Connection>VS Network</epasd:Connection>
2349 <epasd:Description>Virtual NIC</epasd:Description>
2350 <epasd:ElementName>Ethernet Port</epasd:ElementName>
2351
2352 <epasd:InstanceID>3</epasd:InstanceID>
2353 <epasd:NetworkPortProfileID>aaaaaaaa-bbbb-cccc-dddd-
2354 eeeeeeeeeee</epasd:NetworkPortProfileID>
2355 <epasd:NetworkPortProfileIDType>3</epasd:NetworkPortProfileIDType>
2356 <epasd:Reservation>1</epasd:Reservation>
2357 <epasd:ResourceType>10</epasd:ResourceType>
2358 <epasd:VirtualQuantityUnits>1</epasd:VirtualQuantityUnits>
2359 </EthernetPortItem>
2360 <StorageItem>
2361 <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
2362 <sasd:Description>Virtual Disk</sasd:Description>
2363 <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
2364 <sasd:InstanceID>4</sasd:InstanceID>
2365 <sasd:Reservation>100</sasd:Reservation>
2366 <sasd:ResourceType>31</sasd:ResourceType>
2367 <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
2368 </StorageItem>
2369 </VirtualHardwareSection>
2370 <OperatingSystemSection ovf:id="58" ovf:required="false">
2371 <Info>Guest Operating System</Info>
2372 <Description>OS</Description>
2373 </OperatingSystemSection>
2374 </VirtualSystem>
2375 </Envelope>

```

## 2376 E.2 Example 2 (OVF descriptor for one virtual system and one network with a 2377 locally referenced network port profile)

2378 The example below shows an OVF descriptor that describes a virtual system and a network to which it  
2379 connects. The virtual system description in this example uses a network port profile that is described in a  
2380 local file that is contained in the same OVF package. The network described in the network section uses  
2381 the same network port profile description. The network port profile described in this example is used to  
2382 reserve 1 Gbps of bandwidth.

```

2383 <?xml version="1.0" encoding="UTF-8"?>
2384 <Envelope xsi:schemaLocation="http://schemas.dmtf.org/ovf/envelope/2
2385 file:///C:/dsp8023_2.0.0_wgv0.9.5.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2386 xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/2" xmlns="http://schemas.dmtf.org/ovf/envelope/2"
2387 xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
2388 xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
2389 xmlns:epasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2390 schema/2/CIM_EthernetPortAllocationSettingData"
2391 xmlns:sasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_StorageAllocationSettingData">
2392 <!-- References to all external files -->
2393 <References>
2394 <File ovf:id="file1" ovf:href="vmdisk1.vmdk" ovf:size="2000000000"/>
2395 <File ovf:id="networkportprofile1" ovf:href="NetworkPortProfile1.xml"/>
2396 </References>
2397 <!-- Describes meta-information for all virtual disks in the package -->
2398 <DiskSection>
2399 <Info>Describes the set of virtual disks</Info>
2400 <Disk ovf:diskId="vmdisk1" ovf:fileRef="file1" ovf:capacity="4294967296"
2401 ovf:format="http://www.examplecompany.com/interfaces/specifications/vmdk.html#sparse"/>
2402 </DiskSection>
2403 <!-- Describes all networks used in the package -->
2404 <NetworkSection>
2405 <Info>List of logical networks used in the package</Info>
2406 <Network ovf:name="VS Network">
2407 <Description>The network that VSs connect to</Description>
2408 <NetworkPortProfileURI>file:networkportprofile1</NetworkPortProfileURI>
2409 </Network>
2410 </NetworkSection>
2411 <VirtualSystem ovf:id="vm">
2412 <Info>Describes a virtual system</Info>
2413 <Name>Virtual Appliance One</Name>
2414 <ProductSection>
2415 <Info>Describes product information for the appliance</Info>
2416 <Product>The Great Appliance</Product>
2417 <Vendor>Some Great Corporation</Vendor>
2418 <Version>13.00</Version>
2419 <FullVersion>13.00-b5</FullVersion>
2420 <ProductUrl>http://www.somegreatcorporation.com/greatappliance</ProductUrl>
2421 <VendorUrl>http://www.somegreatcorporation.com/</VendorUrl>
2422 <Property ovf:key="adminemail" ovf:type="string">
2423 <Description>Email address of administrator</Description>
2424 </Property>
2425 <Property ovf:key="app_ip" ovf:type="string" ovf:defaultValue="192.168.0.10">
2426 <Description>The IP address of this appliance</Description>
2427 </Property>
2428 </ProductSection>
2429 <AnnotationSection ovf:required="false">
2430 <Info>A random annotation on this service. It can be ignored</Info>
2431 <Annotation>Contact customer support if you have any problems</Annotation>
2432 </AnnotationSection>
2433 <EulaSection>
2434 <Info>License information for the appliance</Info>
2435 <License>Insert your favorite license here</License>
2436 </EulaSection>
2437 <VirtualHardwareSection>
2438 <Info>Memory = 4 GB, CPU = 1 GHz, Disk = 100 GB, 1 Ethernet nic</Info>
2439 <Item>
2440 <rasd:AllocationUnits>Hertz*10^9</rasd:AllocationUnits>
2441 <rasd:Description>Virtual CPU</rasd:Description>

```



```

2442 <rasd:ElementName>1 GHz virtual CPU</rasd:ElementName>
2443 <rasd:InstanceID>1</rasd:InstanceID>
2444 <rasd:Reservation>1</rasd:Reservation>
2445 <rasd:ResourceType>3</rasd:ResourceType>
2446 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2447 </Item>
2448 <Item>
2449 <rasd:AllocationUnits>byte*2^30</rasd:AllocationUnits>
2450 <rasd:Description>Memory</rasd:Description>
2451 <rasd:ElementName>1 GByte of memory</rasd:ElementName>
2452 <rasd:InstanceID>2</rasd:InstanceID>
2453 <rasd:ResourceType>4</rasd:ResourceType>
2454 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2455 </Item>
2456 <EthernetPortItem>
2457 <epasd:Address>00-16-8B-DB-00-5E</epasd:Address>
2458 <epasd:Connection>VS Network</epasd:Connection>
2459 <epasd:Description>Virtual NIC</epasd:Description>
2460 <epasd:ElementName>Ethernet Port</epasd:ElementName>
2461
2462 <epasd:InstanceID>3</epasd:InstanceID>
2463 <epasd:NetworkPortProfileID>file:networkportprofile1</epasd:NetworkPortProfileID>
2464 <epasd:NetworkPortProfileIDType>2</epasd:NetworkPortProfileIDType>
2465 <epasd:ResourceType>10</epasd:ResourceType>
2466 <epasd:VirtualQuantityUnits>1</epasd:VirtualQuantityUnits>
2467 </EthernetPortItem>
2468 <StorageItem>
2469 <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
2470 <sasd:Description>Virtual Disk</sasd:Description>
2471 <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
2472 <sasd:InstanceID>4</sasd:InstanceID>
2473 <sasd:Reservation>100</sasd:Reservation>
2474 <sasd:ResourceType>31</sasd:ResourceType>
2475 <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
2476 </StorageItem>
2477 </VirtualHardwareSection>
2478 <OperatingSystemSection ovf:id="58" ovf:required="false">
2479 <Info>Guest Operating System</Info>
2480 <Description>OS</Description>
2481 </OperatingSystemSection>
2482 </VirtualSystem>
2483 </Envelope>

```

### 2484 E.3 Example 3 (OVF descriptor for one virtual system and one network with a 2485 network port profile referenced by a URI)

2486 The example below shows an OVF descriptor that describes a virtual system and a network to which it  
2487 connects. The virtual system description in this example uses a network port profile that is described by a  
2488 URI. The network described in the network section uses the same network port profile description. The  
2489 network port profile described in this example is used to reserve 1 Gbps of bandwidth.

```

2490 <?xml version="1.0" encoding="UTF-8"?>
2491 <Envelope xsi:schemaLocation="http://schemas.dmtf.org/ovf/envelope/2
2492 file:///C:/dsp8023_2.0.0_wgv0.9.5.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2493 xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/2" xmlns="http://schemas.dmtf.org/ovf/envelope/2"
2494 xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
2495 xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
2496 xmlns:epasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2497 schema/2/CIM_EthernetPortAllocationSettingData"
2498 xmlns:sasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_StorageAllocationSettingData">
2499 <!-- References to all external files -->
2500 <References>
2501 <File ovf:id="file1" ovf:href="vmdisk1.vmdk" ovf:size="2000000000"/>
2502 </References>
2503 <!-- Describes meta-information for all virtual disks in the package -->
2504 <DiskSection>
2505 <Info>Describes the set of virtual disks</Info>

```

```

2506 <Disk ovf:diskId="vmdisk1" ovf:fileRef="file1" ovf:capacity="4294967296"
2507 ovf:format="http://www.examplecompany.com/interfaces/specifications/vmdk.html#sparse"/>
2508 </DiskSection>
2509 <!-- Describes all networks used in the package -->
2510 <NetworkSection>
2511 <Info>List of logical networks used in the package</Info>
2512 <Network ovf:name="VS Network">
2513 <Description>The network that the VSs connect to</Description>
2514
2515 <NetworkPortProfileURI>http://www.dmtf.org/networkportprofiles/networkportprofile1.xml</Netwo
2516 rkPortProfileURI>
2517 </Network>
2518 </NetworkSection>
2519 <VirtualSystem ovf:id="vm">
2520 <Info>Describes a virtual system</Info>
2521 <Name>Virtual Appliance One</Name>
2522 <ProductSection>
2523 <Info>Describes product information for the appliance</Info>
2524 <Product>The Great Appliance</Product>
2525 <Vendor>Some Great Corporation</Vendor>
2526 <Version>13.00</Version>
2527 <FullVersion>13.00-b5</FullVersion>
2528 <ProductUrl>http://www.somegreatcorporation.com/greatappliance</ProductUrl>
2529 <VendorUrl>http://www.somegreatcorporation.com/</VendorUrl>
2530 <Property ovf:key="adminemail" ovf:type="string">
2531 <Description>Email address of administrator</Description>
2532 </Property>
2533 <Property ovf:key="app_ip" ovf:type="string" ovf:defaultValue="192.168.0.10">
2534 <Description>The IP address of this appliance</Description>
2535 </Property>
2536 </ProductSection>
2537 <AnnotationSection ovf:required="false">
2538 <Info>A random annotation on this service. It can be ignored</Info>
2539 <Annotation>Contact customer support if you have any problems</Annotation>
2540 </AnnotationSection>
2541 <EulaSection>
2542 <Info>License information for the appliance</Info>
2543 <License>Insert your favorite license here</License>
2544 </EulaSection>
2545 <VirtualHardwareSection>
2546 <Info>Memory = 4 GB, CPU = 1 GHz, Disk = 100 GB, 1 Ethernet nic</Info>
2547 <Item>
2548 <rasd:AllocationUnits>Hertz*10^9</rasd:AllocationUnits>
2549 <rasd:Description>Virtual CPU</rasd:Description>
2550 <rasd:ElementName>1 GHz virtual CPU</rasd:ElementName>
2551 <rasd:InstanceID>1</rasd:InstanceID>
2552 <rasd:Reservation>1</rasd:Reservation>
2553 <rasd:ResourceType>3</rasd:ResourceType>
2554 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2555 </Item>
2556 <Item>
2557 <rasd:AllocationUnits>byte*2^30</rasd:AllocationUnits>
2558 <rasd:Description>Memory</rasd:Description>
2559 <rasd:ElementName>1 GByte of memory</rasd:ElementName>
2560 <rasd:InstanceID>2</rasd:InstanceID>
2561 <rasd:ResourceType>4</rasd:ResourceType>
2562 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2563 </Item>
2564 <EthernetPortItem>
2565 <epasd:Address>00-16-8B-DB-00-5E</epasd:Address>
2566 <epasd:Connection>VS Network</epasd:Connection>
2567 <epasd:Description>Virtual NIC</epasd:Description>
2568 <epasd:ElementName>Ethernet Port</epasd:ElementName>
2569
2570 <epasd:InstanceID>3</epasd:InstanceID>
2571
2572 <epasd:NetworkPortProfileID>http://www.dmtf.org/networkportprofiles/networkportprofile1.xml</
2573 epasd:NetworkPortProfileID>
2574 <epasd:NetworkPortProfileIDType>2</epasd:NetworkPortProfileIDType>
2575 <epasd:ResourceType>10</epasd:ResourceType>

```

```

2576 <epas:VirtualQuantityUnits>1</epas:VirtualQuantityUnits>
2577 </EthernetPortItem>
2578 <StorageItem>
2579 <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
2580 <sasd:Description>Virtual Disk</sasd:Description>
2581 <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
2582 <sasd:InstanceID>4</sasd:InstanceID>
2583 <sasd:Reservation>100</sasd:Reservation>
2584 <sasd:ResourceType>31</sasd:ResourceType>
2585 <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
2586 </StorageItem>
2587 </VirtualHardwareSection>
2588 <OperatingSystemSection ovf:id="58" ovf:required="false">
2589 <Info>Guest Operating System</Info>
2590 <Description>OS</Description>
2591 </OperatingSystemSection>
2592 </VirtualSystem>
2593 </Envelope>

```

#### 2594 **E.4 Example 4 (OVF descriptor for two virtual systems and one network with** 2595 **two network port profiles referenced by URIs)**

2596 The example below shows an OVF descriptor that describes two virtual systems and a network to which  
2597 they connect. Each virtual system description in this example uses a network port profile that is described  
2598 by a URI. The network described in the network section uses the same two network port profiles. The two  
2599 network port profiles described in this example are used to reserve 1 Gbps of bandwidth and describe  
2600 general network traffic respectively. Annex E.5 and E.6 are examples of these network port profiles.

```

2601 <?xml version="1.0" encoding="UTF-8"?>
2602 <Envelope xsi:schemaLocation="http://schemas.dmtf.org/ovf/envelope/2
2603 file:///C:/dsp8023 2.0.0 wgv0.9.5.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2604 xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/2" xmlns="http://schemas.dmtf.org/ovf/envelope/2"
2605 xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
2606 xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
2607 xmlns:epasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2608 schema/2/CIM_EthernetPortAllocationSettingData"
2609 xmlns:sasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_StorageAllocationSettingData">
2610 <!-- References to all external files -->
2611 <References>
2612 <File ovf:id="file1" ovf:href="vmdisk1.vmdk" ovf:size="2000000000"/>
2613 </References>
2614 <!-- Describes meta-information for all virtual disks in the package -->
2615 <DiskSection>
2616 <Info>Describes the set of virtual disks</Info>
2617 <Disk ovf:diskId="vmdisk1" ovf:fileRef="file1" ovf:capacity="4294967296"
2618 ovf:format="http://www.examplecompany.com/interfaces/specifications/vmdk.html#sparse"/>
2619 </DiskSection>
2620 <!-- Describes all networks used in the package -->
2621 <NetworkSection>
2622 <Info>List of logical networks used in the package</Info>
2623 <Network ovf:name="VS Network">
2624 <Description>The network that the VSs connect to</Description>
2625 <!-- Network port profile for storage traffic -->
2626
2627 <NetworkPortProfileURI>http://www.dmtf.org/networkportprofiles/networkportprofile1.xml</Netwo
2628 rkPortProfileURI>
2629 <!-- Network port profile for networking traffic -->
2630
2631 <NetworkPortProfileURI>http://www.dmtf.org/networkportprofiles/networkportprofile2.xml</Netwo
2632 rkPortProfileURI>
2633 </Network>
2634 </NetworkSection>
2635 <VirtualSystemCollection ovf:id="vscl">
2636 <Info>Collection of 2 VSs</Info>
2637 <VirtualSystem ovf:id="storage server">
2638 <Info>Describes a virtual system</Info>
2639 <Name>Virtual Appliance One</Name>
2640 <ProductSection>

```

```

2641 <Info>Describes product information for the appliance</Info>
2642 <Product>The Great Appliance</Product>
2643 <Vendor>Some Great Corporation</Vendor>
2644 <Version>13.00</Version>
2645 <FullVersion>13.00-b5</FullVersion>
2646 <ProductUrl>http://www.somegreatcorporation.com/greatappliance</ProductUrl>
2647 <VendorUrl>http://www.somegreatcorporation.com/</VendorUrl>
2648 <Property ovf:key="adminemail" ovf:type="string">
2649 <Description>Email address of administrator</Description>
2650 </Property>
2651 <Property ovf:key="app_ip" ovf:type="string" ovf:defaultValue="192.168.0.10">
2652 <Description>The IP address of this appliance</Description>
2653 </Property>
2654 </ProductSection>
2655 <AnnotationSection ovf:required="false">
2656 <Info>A random annotation on this service. It can be ignored</Info>
2657 <Annotation>Contact customer support if you have any problems</Annotation>
2658 </AnnotationSection>
2659 <EulaSection>
2660 <Info>License information for the appliance</Info>
2661 <License>Insert your favorite license here</License>
2662 </EulaSection>
2663 <VirtualHardwareSection>
2664 <Info>Memory = 4 GB, CPU = 1 GHz, Disk = 100 GB, 1 Ethernet nic</Info>
2665 <Item>
2666 <rasd:AllocationUnits>Hertz*10^9</rasd:AllocationUnits>
2667 <rasd:Description>Virtual CPU</rasd:Description>
2668 <rasd:ElementName>1 GHz virtual CPU</rasd:ElementName>
2669 <rasd:InstanceID>1</rasd:InstanceID>
2670 <rasd:Reservation>1</rasd:Reservation>
2671 <rasd:ResourceType>3</rasd:ResourceType>
2672 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2673 </Item>
2674 <Item>
2675 <rasd:AllocationUnits>byte*2^30</rasd:AllocationUnits>
2676 <rasd:Description>Memory</rasd:Description>
2677 <rasd:ElementName>1 GByte of memory</rasd:ElementName>
2678 <rasd:InstanceID>2</rasd:InstanceID>
2679 <rasd:ResourceType>4</rasd:ResourceType>
2680 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2681 </Item>
2682 <EthernetPortItem>
2683 <epasd:Address>00-16-8B-DB-00-5E</epasd:Address>
2684 <epasd:Connection>VS Network</epasd:Connection>
2685 <epasd:Description>Virtual NIC</epasd:Description>
2686
2687 <epasd:ElementName>Ethernet Port</epasd:ElementName>
2688
2689 <epasd:InstanceID>3</epasd:InstanceID>
2690
2691 <epasd:NetworkPortProfileID>http://www.dmtf.org/networkportprofiles/networkportprofile1.xml</
2692 epasd:NetworkPortProfileID>
2693 <epasd:NetworkPortProfileIDType>2</epasd:NetworkPortProfileIDType>
2694 <epasd:ResourceType>10</epasd:ResourceType>
2695 <epasd:VirtualQuantityUnits>1</epasd:VirtualQuantityUnits>
2696 </EthernetPortItem>
2697 <StorageItem>
2698 <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
2699 <sasd:Description>Virtual Disk</sasd:Description>
2700 <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
2701 <sasd:InstanceID>4</sasd:InstanceID>
2702 <sasd:Reservation>100</sasd:Reservation>
2703 <sasd:ResourceType>31</sasd:ResourceType>
2704 <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
2705 </StorageItem>
2706 </VirtualHardwareSection>
2707 <OperatingSystemSection ovf:id="58" ovf:required="false">
2708 <Info>Guest Operating System</Info>
2709 <Description>OS</Description>
2710 </OperatingSystemSection>

```

```

2711 </VirtualSystem>
2712 <VirtualSystem ovf:id="web-server">
2713 <Info>Describes a virtual system</Info>
2714 <Name>Virtual Appliance Two</Name>
2715 <ProductSection>
2716 <Info>Describes product information for the appliance</Info>
2717 <Product>The Great Appliance</Product>
2718 <Vendor>Some Great Corporation</Vendor>
2719 <Version>13.00</Version>
2720 <FullVersion>13.00-b5</FullVersion>
2721 <ProductUrl>http://www.somegreatcorporation.com/greatappliance</ProductUrl>
2722 <VendorUrl>http://www.somegreatcorporation.com/</VendorUrl>
2723 <Property ovf:key="adminemail" ovf:type="string">
2724 <Description>Email address of administrator</Description>
2725 </Property>
2726 <Property ovf:key="app ip" ovf:type="string" ovf:defaultValue="192.168.0.10">
2727 <Description>The IP address of this appliance</Description>
2728 </Property>
2729 </ProductSection>
2730 <AnnotationSection ovf:required="false">
2731 <Info>A random annotation on this service. It can be ignored</Info>
2732 <Annotation>Contact customer support if you have any problems</Annotation>
2733 </AnnotationSection>
2734 <EulaSection>
2735 <Info>License information for the appliance</Info>
2736 <License>Insert your favorite license here</License>
2737 </EulaSection>
2738 <VirtualHardwareSection>
2739 <Info>Memory = 4 GB, CPU = 1 GHz, Disk = 100 GB, 1 Ethernet nic</Info>
2740 <Item>
2741 <rasd:AllocationUnits>Hertz*10^9</rasd:AllocationUnits>
2742 <rasd:Description>Virtual CPU</rasd:Description>
2743 <rasd:ElementName>1 GHz virtual CPU</rasd:ElementName>
2744 <rasd:InstanceID>1</rasd:InstanceID>
2745 <rasd:Reservation>1</rasd:Reservation>
2746 <rasd:ResourceType>3</rasd:ResourceType>
2747 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2748 </Item>
2749 <Item>
2750 <rasd:AllocationUnits>byte*2^30</rasd:AllocationUnits>
2751 <rasd:Description>Memory</rasd:Description>
2752 <rasd:ElementName>1 GByte of memory</rasd:ElementName>
2753 <rasd:InstanceID>2</rasd:InstanceID>
2754 <rasd:ResourceType>4</rasd:ResourceType>
2755 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2756 </Item>
2757 <EthernetPortItem>
2758 <epasd:Address>00-16-8B-DB-00-5F</epasd:Address>
2759 <epasd:Connection>VS Network</epasd:Connection>
2760 <epasd:Description>Virtual NIC</epasd:Description>
2761
2762 <epasd:ElementName>Ethernet Port</epasd:ElementName>
2763 <!-- Virtual NIC for networking traffic -->
2764 <epasd:InstanceID>3</epasd:InstanceID>
2765
2766 <epasd:NetworkPortProfileID>http://www.dmtf.org/networkportprofiles/networkportprofile2.xml</
2767 epasd:NetworkPortProfileID>
2768 <epasd:NetworkPortProfileIDType>2</epasd:NetworkPortProfileIDType>
2769 <epasd:ResourceType>10</epasd:ResourceType>
2770 <epasd:VirtualQuantityUnits>1</epasd:VirtualQuantityUnits>
2771 </EthernetPortItem>
2772 <StorageItem>
2773 <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
2774 <sasd:Description>Virtual Disk</sasd:Description>
2775 <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
2776 <sasd:InstanceID>4</sasd:InstanceID>
2777 <sasd:Reservation>100</sasd:Reservation>
2778 <sasd:ResourceType>31</sasd:ResourceType>
2779 <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
2780 </StorageItem>

```

```

2781 </VirtualHardwareSection>
2782 <OperatingSystemSection ovf:id="58" ovf:required="false">
2783 <Info>Guest Operating System</Info>
2784 <Description>OS</Description>
2785 </OperatingSystemSection>
2786 </VirtualSystem>
2787 </VirtualSystemCollection>
2788 </Envelope>

```

## 2789 E.5 Example 5 (networkportprofile1.xml)

2790  
2791 Network port profile example for bandwidth reservation.

```

2792 <?xml version="1.0" encoding="UTF-8"?>
2793 <NetworkPortProfile xsi:schemaLocation="http://schemas.dmtf.org/ovf/networkportprofile/1
2794 http://schemas.dmtf.org/ovf/networkportprofile/1/dsp8049.xsd"
2795 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2796 xmlns="http://schemas.dmtf.org/ovf/networkportprofile/1"
2797 xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
2798 xmlns:epasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2799 schema/2/CIM_EthernetPortAllocationSettingData">
2800 <Item>
2801 <epasd:AllocationUnits>bit / second * 10^9</epasd:AllocationUnits>
2802 <epasd:ElementName>Network Port Profile 1</epasd:ElementName>
2803 <epasd:InstanceID>1</epasd:InstanceID>
2804 <epasd:NetworkPortProfileID>aaaaaaaa-bbbb-cccc-dddd-
2805 eeeeeeeeeee</epasd:NetworkPortProfileID>
2806 <epasd:NetworkPortProfileIDType>3</epasd:NetworkPortProfileIDType>
2807 <epasd:Reservation>1</epasd:Reservation>
2808 </Item>
2809 </NetworkPortProfile>

```

## 2810 E.6 Example 6 (networkportprofile2.xml)

2811  
2812 Network port profile example showing priority setting.

```

2813 <?xml version="1.0" encoding="UTF-8"?>
2814 <NetworkPortProfile xsi:schemaLocation="http://schemas.dmtf.org/ovf/networkportprofile/1
2815 http://schemas.dmtf.org/ovf/networkportprofile/1/dsp8049.xsd"
2816 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2817 xmlns="http://schemas.dmtf.org/ovf/networkportprofile/1"
2818 xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
2819 xmlns:epasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2820 schema/2/CIM_EthernetPortAllocationSettingData">
2821 <Item>
2822 <epasd:AllowedPriorities>0</epasd:AllowedPriorities>
2823 <epasd:AllowedPriorities>1</epasd:AllowedPriorities>
2824 <epasd:DefaultPriority>0</epasd:DefaultPriority>
2825 <epasd:ElementName>Network Port Profile 2</epasd:ElementName>
2826 <epasd:InstanceID>2</epasd:InstanceID>
2827 <epasd:NetworkPortProfileID>aaaaaaaa-bbbb-cccc-dddd-
2828 ffffffff</epasd:NetworkPortProfileID>
2829 <epasd:NetworkPortProfileIDType>3</epasd:NetworkPortProfileIDType>
2830 </Item>
2831 </NetworkPortProfile>

```

## ANNEX F (informative)

2832  
2833  
2834  
2835

### Deployment considerations

2836 This standard defines an OVF package and the main clauses in this standard deal with this subject  
2837 matter. However, there are deployment considerations necessary to meet the expectations of the OVF  
2838 package author. These are listed below.

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

2840 A deployment function shall verify the ovf package signature and should validate the certificate.

#### F.2 Virtual hardware deployment considerations

2842 If there are multiple virtual hardware sections, the deployment function should select the most appropriate  
2843 one for the target virtualization platform.

2844 If no backing is specified for a device that requires a backing, the deployment function shall make an  
2845 appropriate choice, for example, by prompting the user. More than one backing for a device shall not be  
2846 specified.

2847 The deployment function should select the normal value for a resource allocation but may adjust it within  
2848 the specified range. The virtualization management may further alter the resource allocation within the  
2849 specified range for performance tuning.

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

2851 The sharing of disk blocks at runtime is optional and virtualization platform specific and shall not be visible  
2852 to the guest software.

2853 A virtualization platform may share storage extents to minimize the amount of space required to support  
2854 the virtual systems. If storage extents are shared by the virtualization platform, this sharing is not visible to  
2855 the guest software.

2856 If present, the AnnotationSection element may be displayed during deployment of the OVF package.

2857 If present, the EULASection(s) shall be displayed and accepted during deployment of an OVF package. If  
2858 automated deployment is used, the deployment function shall have a methodology to provide implicit  
2859 acceptance.

2860 If virtual disks or other files are included by reference, the deployment function shall acquire those files  
2861 prior to the virtual system being launched.

2862 If the specified boot source is a device type, the deployment function should try all the devices of that  
2863 device type specified.

**ANNEX G  
(informative)****Change log**2864  
2865  
2866  
2867

| Version | Date       | Description                                                                                                    |
|---------|------------|----------------------------------------------------------------------------------------------------------------|
| 1.0.0   | 2009-02-22 | DMTF Standard release                                                                                          |
| 1.1.0   | 2010-01-12 | DMTF Standard release                                                                                          |
| 2.0.0   | 2012-10-29 | DMTF Standard release                                                                                          |
| 2.1.0   | 2013-12-12 | DMTF Standard release                                                                                          |
| 2.1.1   | 2015-08-27 | DMTF Standard release<br>Errata to address ANSI Editor comments; see July 10, 2015 OVF WG minutes for details. |

2868