



1
2
3
4

Document Number: DSP0243

Date: 2013-05-29

Version: 2.1.0a

5 **Open Virtualization Format Specification**

Information for Work-in-Progress version:

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

It expires on:2013-10-31

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

6 **Document Type: Specification**
7 **Document Status: Work in Progress**
8 **Document Language: en-US**

9 Copyright Notice

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

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

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25 withdrawn or modified after publication, and shall be indemnified and held harmless by any party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

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

CONTENTS

32	Foreword	6
33	Introduction.....	8
34	1 Scope	9
35	2 Normative References.....	9
36	3 Terms and Definitions	10
37	4 Symbols and Abbreviated Terms	12
38	5 OVF Package	12
39	5.1 OVF Package Structure	12
40	5.2 Virtual Disk Formats.....	14
41	5.3 OVF PackageOptions	14
42	5.4 Distribution as a Set of Files	15
43	6 OVF Descriptor.....	15
44	7 Envelope Element	15
45	7.1 File References	16
46	7.2 Content Element	17
47	7.3 Extensibility	17
48	7.4 Conformance	18
49	8 Virtual Hardware Description.....	18
50	8.1 VirtualHardwareSection	18
51	8.2 Extensibility	19
52	8.3 Virtual Hardware Elements	20
53	8.4 Ranges on Elements.....	22
54	9 Core Metadata Sections in version 2	24
55	9.1 DiskSection	25
56	9.2 NetworkSection.....	26
57	9.3 ResourceAllocationSection	26
58	9.4 AnnotationSection.....	27
59	9.5 ProductSection.....	27
60	9.5.1 Property Elements	28
61	9.6 EulaSection.....	30
62	9.7 StartupSection	30
63	9.8 DeploymentOptionSection	31
64	9.9 OperatingSystemSection	32
65	9.10 InstallSection.....	32
66	9.11 EnvironmentFilesSection	33
67	9.12 BootDeviceSection.....	33
68	9.13 SharedDiskSection	34
69	9.14 ScaleOutSection	34
70	9.15 PlacementGroupSection and PlacementSection.....	35
71	9.16 Encryption Section	36
72	10 Internationalization	36
73	10.1 Internal Resource Bundles	37
74	10.2 External Resource Bundles	37
75	10.3 Message Content in External File.....	37
76	11 OVF Environment and OVF Environment File	37
77	11.1 Transport Media.....	38
78	11.2 Transport Media Type.....	39
79	ANNEX A (informative) Symbols and Conventions	40
80	ANNEX B (normative) OVF XSD	41
81	ANNEX C (informative) OVF Mime Type Registration Template	42

82	ANNEX D (informative) OVF Examples	44
83	D.1 Examples of OVF Package Structure	44
84	D.2 Examples of Distribution of Files	44
85	D.3 Example of Envelope Element.....	45
86	D.4 Example of File References.....	46
87	D.5 Example of Content Element	46
88	D.6 Examples of Extensibility	46
89	D.7 Examples of VirtualHardwareSection	47
90	D.8 Examples of Virtual Hardware Elements	48
91	D.9 Example of Ranges on Elements	48
92	D.10 Example of DiskSection	49
93	D.11 Example of NetworkSection.....	49
94	D.12 Example of ResourceAllocationSection.....	50
95	D.13 Example of Annotation.....	50
96	D.14 Example of Product Section	50
97	D.15 Example of EULA Section	51
98	D.16 Example of DeploymentOptionSection	51
99	D.17 Example of OperatingSystemSection	52
100	D.18 Example of InstallSection	52
101	D.19 Example of EnvironmentFilesSection	53
102	D.20 Example of BootDeviceSection	53
103	D.21 Example of SharedDiskSection	54
104	D.22 Example of ScaleOutSection	54
105	D.23 Example of PlacementGroupSection.....	55
106	D.24 Example of EncryptionSection.....	56
107	D.25 Example of Internationalization.....	57
108	D.26 Example of Message Content in an External File.....	58
109	D.27 Example of Environment Document	59
110	ANNEX E (informative) Network Port Profile Examples	60
111	E.1 Example 1 (OVF Descriptor for One Virtual System and One Network with an Inlined Network Port Profile).....	60
113	E.2 Example 2 (OVF Descriptor for One Virtual System and One Network with a Locally Referenced Network Port Profile)	62
115	E.3 Example 3 (OVF Descriptor for One Virtual System and One Network with a Network Port Profile referenced by a URI)	63
117	E.4 Example 4 (OVF Descriptor for Two Virtual Systems and One Network with Two Network Port Profiles referenced by URIs)	65
119	E.5 Example 5 (networkportprofile1.xml)	68
120	E.6 Example 6 (networkportprofile2.xml)	68
121	ANNEX F (informative) Deployment Considerations	69
122	F.1 OVF Package Structure Deployment Considerations	69
123	F.2 Virtual Hardware Deployment Considerations.....	69
124	F.3 Core Metadata Sections Deployment Considerations.....	69
125	ANNEX G (informative) Bibliography	70
126	ANNEX H (informative) Change Log	71
127		

128 **Tables**

129 Table 1 – XML Namespace Prefixes 15

130 Table 2 – Actions for Child Elements with `ovf:required` Attribute..... 20

131 Table 3 – HostResource Element 21

132 Table 4 – Elements for Virtual Devices and Controllers 22

133 Table 5 – Core Metadata Sections 24

134 Table 6 – Property Types 29

135 Table 7 – Property Qualifiers 29

136 Table 8 – Core Sections..... 38

137

138

Foreword

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

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

143

144	Lawrence Lamers	VMware Inc. (Chair)
145	Hemal Shah	Broadcom Corporation (co-Editor)
146	Steffen Grarup	VMware Inc. (co-Editor)
147		
148	Vincent Kowalski	BMC Software
149	Hemal Shah	Broadcom Corporation
150	John Crandall	Brocade Communications Systems
151	Marvin Waschke	CA Technologies
152	Naveen Joy	Cisco
153	Steven Neely	Cisco
154	Shishir Pardikar	Citrix Systems Inc.
155	Thomas Root	Citrix Systems Inc.
156	Richard Landau	DMTF Fellow
157	Jacques Durand	Fujitsu
158	Derek Coleman	Hewlett-Packard Company
159	Robert Freund	Hitachi, Ltd.
160	Fred Maciel	Hitachi, Ltd.
161	Eric Wells	Hitachi, Ltd.
162	Abdellatif Touimi	Huawei
163	Jeff Wheeler	Huawei
164	HengLiang Zhang	Huawei
165	Oliver Benke	IBM
166	Ron Doyle	IBM
167	Michael Gering	IBM
168	Michael Johanssen	IBM
169	Andreas Maier	IBM
170	Marc-Arthur Pierre-Louis	IBM
171	John Leung	Intel Corporation
172	Nitin Bhat	Microsoft Corporation
173	Maurizio Carta	Microsoft Corporation
174	Monica Martin	Microsoft Corporation
175	John Parchem	Microsoft Corporation
176	Ed Reed	Microsoft Corporation
177	Nihar Shah	Microsoft Corporation
178	Cheng Wei	Microsoft Corporation
179	Narayan Venkat	NetApp
180	Tatyana Bagerman	Oracle
181	Srinivas Maturi	Oracle
182	Dr. Fermín Galán Márquez	Telefónica
183	Miguel Ángel Peñalvo	Telefónica
184	Dr. Fernando de la Iglesia	Telefónica
185	Álvaro Polo	Telefónica
186	Steffen Grarup	VMware Inc.
187	Lawrence Lamers	VMware Inc.
188	Rene Schmidt	VMware Inc.
189	Paul Ferdinand	WBEM Solutions

190	Junsheng Chu	ZTE Corporation
191	Bhumip Khasnabish	ZTE Corporation
192	Ghazanfar Ali	ZTE Corporation

193

Introduction

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

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

198 • **Optimized for distribution**

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

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

202 OVF supports validation of the entire package and each virtual machine or metadata
203 component of the OVF during the installation phases of the virtual machine (VM) lifecycle
204 management process. It also packages with the package relevant user-readable descriptive
205 information that a virtualization platform can use to streamline the installation experience.

206 • **Supports both single VM and multiple-VM configurations**

207 OVF supports both standard single VM packages and packages containing complex, multi-tier
208 services consisting of multiple interdependent VMs.

209 • **Portable VM packaging**

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

214 • **Vendor and platform independent**

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

217 • **Extensible**

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

221 • **Localizable**

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

225 • **Open standard**

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

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

230

Open Virtualization Format Specification

231 1 Scope

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

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

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

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

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

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

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

248 2 Normative References

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

252 [ISO/IEC/IEEE 9945:2009](#): Information technology -- Portable Operating System Interface (POSIX®) Base
253 Specifications, Issue 7

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

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

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

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

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

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

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

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

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

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

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

265 DMTF DSP8023, *Open Virtualization Format (OVF) 2 XML Schema*,

266 <http://schemas.dmtf.org/ovf/envelope/2/dsp8023.xsd>

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

291 **3 Terms and Definitions**

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

293 **3.1**

294 **authoring function**

295 the creation of the OVF package

296 **3.2**

297 **can**

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

299 **3.3**

300 **cannot**

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

302 **3.4**

303 **conditional**

304 indicates requirements to be followed strictly to conform to the document when the specified conditions
305 are met

- 306 **3.5**
307 **mandatory**
308 indicates requirements to be followed strictly to conform to the document and from which no deviation is
309 permitted
- 310 **3.6**
311 **may**
312 indicates a course of action permissible within the limits of the document
- 313 **3.7**
314 **need not**
315 indicates a course of action permissible within the limits of the document
- 316 **3.8**
317 **optional**
318 indicates a course of action permissible within the limits of the document
- 319 **3.9**
320 **shall**
321 indicates requirements to be followed strictly to conform to the document and from which no deviation is
322 permitted
- 323 **3.10**
324 **shall not**
325 indicates requirements to be followed strictly to conform to the document and from which no deviation is
326 permitted
- 327 **3.11**
328 **should**
329 indicates that among several possibilities, one is recommended as particularly suitable, without
330 mentioning or excluding others, or that a certain course of action is preferred but not necessarily required
- 331 **3.12**
332 **should not**
333 indicates that a certain possibility or course of action is deprecated but not prohibited
- 334 **3.13**
335 **deployment function**
336 a function the result of which is a prepared virtual machine.
- 337 **3.14**
338 **guest software**
339 the software that runs inside a virtual system
- 340 **3.15**
341 **OVF package**
342 a single compressed file or a set of files that contains the OVF descriptor file and may contain associated
343 virtual disks, operational metadata, and other files.
- 344 **3.16**
345 **OVF descriptor**
346 an XML file that validates to DSP8023 and provides the information needed to deploy the OVF package
- 347 **3.17**

348 virtualization platform

349 the hypervisor that the virtual systems run on

350 3.18**351 virtual appliance**

352 a service delivered as a software stack that utilizes one or more virtual systems

353 3.19**354 virtual hardware**

355 the processor, memory and I/O resources provided by a virtualization platform that supports a virtual
356 system

357 3.20**358 virtual machine**

359 as defined in System Virtualization Profile

360 3.21**361 virtual system**

362 a virtual machine and the guest software

363 3.22**364 virtual system collection**

365 a collection of virtual systems

366 3.23**367 virtualization management**

368 the software that performs resource allocation and management of virtual systems

369 4 Symbols and Abbreviated Terms

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

371 4.1**372 CIM**

373 Common Information Model

374 4.2**375 IP**

376 Internet Protocol

377 4.3**378 OVF**

379 Open Virtualization Format

380 4.4**381 VM**

382 Virtual Machine

383 5 OVF Package**384 5.1 OVF Package Structure**

385 An OVF package shall consist of the following files:

- 386 • one OVF descriptor with extension `.ovf`
- 387 • zero or one OVF manifest with extension `.mf`
- 388 • zero or one OVF certificate with extension `.cert`
- 389 • zero or more disk image files
- 390 • zero or more additional resource files, such as ISO images

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

392 An OVF package can be stored as either a single compressed file (`.ova`) or a set of files, as described in
393 5.3 and 5.4. Both modes shall be supported.

394 An OVF package may have a manifest file containing the SHA digests of individual files in the package.
395 OVF packages authored according to this version of the specification shall use SHA256 digests; older
396 OVF packages are allowed to use SHA1. The manifest file shall have an extension `.mf` and the same
397 base name as the `.ovf` file and be a sibling of the `.ovf` file. If the manifest file is present, a consumer of
398 the OVF package shall verify the digests by computing the actual SHA digests and comparing them with
399 the digests listed in the manifest file. The manifest file shall contain SHA digests for all distinct files
400 referenced in the `References` element of the OVF descriptor and for no other files. See clause 7.1

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

402 The format of the manifest file is as follows:

```
403 manifest_file = *( file_digest )
404 file_digest  = algorithm "(" file_name ")" "=" sp digest nl
405 algorithm    = "SHA1" | "SHA256"
406 digest       = *( hex-digit )
407 hex-digit    = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" | "a" |
408 "b" | "c" | "d" | "e" | "f"
409 sp           = %x20
410 nl          = %x0A
```

411 See D.1 for an example.

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

415 See ANNEX F for deployment considerations.

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

```
417 certificate_file = manifest_digest certificate_part
418 manifest_digest = algorithm "(" file_name ")" "=" sp signed_digest nl
419 algorithm       = "SHA1" | "SHA256"
420 signed_digest   = *( hex-digit )
421 certificate_part = certificate_header certificate_body certificate_footer
422 certificate_header = "-----BEGIN CERTIFICATE-----" nl
423 certificate_footer = "-----END CERTIFICATE-----" nl
424 certificate_body  = base64-encoded-certificate nl
425                   ; base64-encoded-certificate is a base64-encoded X.509
426                   ; certificate, which may be split across multiple lines
427 hex-digit       = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" | "a"
428 | "b" | "c" | "d" | "e" | "f"
429 sp              = %x20
430 nl              = %x0A
```

431 See D.1 for an example.

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

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

439 5.2 Virtual Disk Formats

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

446 5.3 OVF PackageOptions

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

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

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

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

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

456 1) OVF descriptor
457 2) The remaining files shall be in the same order as listed in the References section (see
458 7.1). Note that any external string resource bundle files for internationalization shall be first in the
459 References section (see clause 10).

460 or

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

467 or

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

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

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

479 **5.4 Distribution as a Set of Files**

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

481 **6 OVF Descriptor**

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

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

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

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

491 **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

492 **7 Envelope Element**

493 The `Envelope` element describes all metadata for the virtual machines (including virtual hardware), as
 494 well as the structure of the OVF package itself.

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

- 496 • A version indication, defined by the XML namespace URIs.
- 497 • A list of file references to all external files that are part of the OVF package, defined by the
 498 `References` element and its `File` child elements, e.g., virtual disk files, ISO images, and
 499 internationalization resources.
- 500 • A metadata part, defined by section elements, defined in clause 9.

- 501 • A description of the content, either a single virtual machine (`VirtualSystem` element) or a
502 collection of multiple virtual machines (`VirtualSystemCollection` element).
- 503 • A specification of message resource bundles for zero or more locales, defined by a `Strings`
504 element for each locale.

505 See D.3 for an example.

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

509 7.1 File References

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

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

515 Each `File` element in the reference part shall be given an identifier using the `ovf:id` attribute. The
516 identifier shall be unique inside an OVF package. Each `File` element shall be specified using the
517 `ovf:href` attribute, that shall contain a URL. Relative-path references and the URL schemes "file",
518 "http", and "https" shall be supported, (see [RFC1738](#) and [RFC3986](#)). Other URL schemes should
519 not be used. If no URL scheme is specified, the value of the `ovf:href` attribute shall be interpreted as a
520 path name of the referenced file relative to the location of the OVF descriptor itself. The relative path
521 name shall use the syntax of relative-path references in [RFC3986](#). The referenced file shall exist. Two
522 different `File` elements shall not reference the same file with their `ovf:href` attributes.

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

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

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

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

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

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

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

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

554 7.2 Content Element

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

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

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

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

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

572 7.3 Extensibility

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

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

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

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

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

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

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

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

594 See D.6 for an example.

595 7.4 Conformance

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

- 598 • the OVF descriptor uses only sections and elements and attributes that are defined in this
599 specification.
600 Conformance Level: 1.
- 601 • the OVF descriptor uses custom sections or elements or attributes that are not defined in this
602 specification and all such extensions are optional as defined in 7.3.
603 Conformance Level: 2.
- 604 • the OVF descriptor uses custom sections or elements that are not defined in this specification
605 and at least one such extension is required as defined in 7.3. The definition of all required
606 extensions shall be publicly available in an open and unencumbered XML Schema. The complete
607 specification may be inclusive in the XML schema or available as a separate document.
608 Conformance Level: 3.

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

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

612 8 Virtual Hardware Description

613 8.1 VirtualHardwareSection

614 The `VirtualHardwareSection` element can be used to describe the virtual hardware used by the virtual
615 system.

616 This standard allows incomplete virtual hardware descriptions.

617 The virtualization platform may create additional virtual hardware devices.

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

619

620 This virtual hardware description is based on the CIM classes `CIM_VirtualSystemSettingData`,
621 `CIM_ResourceAllocationSettingData`, `CIM_EthernetPortAllocationSettingData`, and
622 `CIM_StorageAllocationSettingData`. The XML representation of the CIM model is based on the
623 WS-CIM mapping (DSP0230).

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

626 A `VirtualSystem` element shall have a `VirtualHardwareSection` direct child element. The
627 `VirtualHardwareSection` shall not be a direct child element of a `VirtualSystemCollection`
628 element or of an `Envelope` element.

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

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

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

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

- 639 • zero or one `System` elements
- 640 • zero or more `Item` elements
- 641 • zero or more `EthernetPortItem` elements
- 642 • zero or more `StorageItem` elements.

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

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

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

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

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

664 8.2 Extensibility

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

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

671 **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

672 8.3 Virtual Hardware Elements

673 The element type of the `Item` element in a `VirtualHardwareSection` element is
 674 `CIM_ResourceAllocationSettingData_Type` as defined in `CIM_ResourceAllocationSettingData` .
 675 See ANNEX B.

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

680 The element type of the `EthernetPortItem` element in a `VirtualHardwareSection` element is
 681 `CIM_EthernetPortAllocationSettingData_Type` as defined in
 682 `CIM_EthernetPortAllocationSettingData`. See ANNEX B.

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

687 The element type of the `StorageItem` element in a `VirtualHardwareSection` element is
 688 `CIM_StorageAllocationSettingData_Type` as defined in `CIM_StorageAllocationSettingData`. See
 689 ANNEX B

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

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

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

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

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

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

711 **Table 3 – HostResource Element**

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

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

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

716

Table 4 – Elements for Virtual Devices and Controllers

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

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

720 8.4 Ranges on Elements

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

724 See ANNEX F for virtual hardware deployment considerations.

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

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

733 The semantics of range markers are as follows:

- 734 • InstanceID and ResourceType shall be specified, and the ResourceType shall match
735 other Item elements with the same InstanceID.
- 736 • No more than one min range marker and no more than one max range marker for a given
737 RASD, EPASD, or SASD (identified with InstanceID) shall be specified.
- 738 • An Item, EthernetPortItem, or StorageItem element with a range marker shall have
739 a corresponding Item, EthernetPortItem, or StorageItem element without a range
740 marker, that is, an Item, EthernetPortItem, and StorageItem element with no
741 ovf:bound attribute or ovf:bound attribute with value normal. This corresponding item
742 specifies the default value.
- 743 • For an Item, EthernetPortItem, and StorageItem element where only a min range
744 marker is specified, the max value is unbounded upwards within the set of valid values for the
745 property.
- 746 • For an Item, EthernetPortItem, and StorageItem where only a max range marker is
747 specified, the min value is unbounded downwards within the set of valid values for the property.
- 748 • The default value shall be inside the range.
- 749 • Non-integer elements shall not be used in the range markers for RASD, EPASD, or SASD.
- 750 See D.9 for an example.
- 751

752 **9 Core Metadata Sections in version 2**753 Table 5 shows the core metadata sections that are defined in the `ovf` namespace.754 **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 machine 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 machine 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 machine	VirtualSystem	Zero or one
<code>InstallSection</code> Specifies that the virtual machine 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 machine	VirtualSystem	Zero or more
<code>SharedDiskSection</code> Specifies virtual disks shared by more than one VirtualSystems at runtime	Envelope	Zero or one
<code>ScaleOutSection</code> Specifies that a VirtualSystemCollection contain a set of children that are homogeneous with respect to a prototype	VirtualSystemCollection	Zero or more
<code>PlacementGroupSection</code> Specifies a placement policy for a group of VirtualSystems or VirtualSystemCollections	Envelope	Zero or more
<code>PlacementSection</code> Specifies membership of a particular placement policy group	VirtualSystem VirtualSystemCollection	Zero or one
<code>EncryptionSection</code>	Envelope	Zero or one

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

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

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

760 9.1 DiskSection

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

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

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

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

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

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

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

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

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

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

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

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

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

810 9.2 NetworkSection

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

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

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

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

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

825 Examples of using the network port profiles are in 11.2ANNEX E.

826 9.3 ResourceAllocationSection

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

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

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

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

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

837 9.4 AnnotationSection

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

840 The `AnnotationSection` element is a valid element for the `VirtualSystem` and the
841 `VirtualSystemCollection` entities.

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

843 9.5 ProductSection

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

846 Zero or more `e` elements may be specified within a `VirtualSystem` element or
847 `VirtualSystemCollection` element.

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

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

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

858 See D.14 for an example.

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

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

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

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

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

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

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

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

869 9.5.1 Property Elements

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

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

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

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

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

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

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

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

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

904 The `id` and the value of the `property` elements are exposed to the guest software using the OVF
905 environment file, as described in clause 9.5.1. The value of the `ovfenv:key` attribute of a `Property`
906 element exposed in the OVF environment shall be constructed from the value of the `ovf:key` attribute of
907 the corresponding `Property` element defined in a `ProductSection` entity of an OVF descriptor as
908 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
913 `class-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
917 the `ProductSection` entity. The production `[". " instance-value]` shall be present if and only
918 if `instance-value` is not the empty string.

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

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

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

929

Table 6 – Property Types

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

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

934

Table 7 – Property Qualifiers

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

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

935 9.6 EulaSection

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

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

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

943 9.7 StartupSection

944 The `StartupSection` specifies how a collection of virtual systems identified by a
 945 `VirtualSystemCollection` element is powered on and off. See A.1 for an example.

946 Each `VirtualSystem` element or `VirtualSystemCollection` element that is a direct child of a
 947 `VirtualSystemCollection` element may shall have a corresponding `Item` element in the
 948 `StartupSection` element.. Note that `Item` elements may correspond to both `VirtualSystem` and
 949 `VirtualSystemCollection` entities.

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

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

- 958 • `ovf:id` shall match the value of the `ovf:id` attribute of a `Content` element which is a direct
 959 child of this `VirtualSystemCollection`. That `Content` element describes the virtual
 960 machine or virtual machine collection to which the actions defined in the `Item` element apply.
- 961 • `ovf:order` specifies the startup order using non-negative integer values. If the `ovf:order`
 962 ="0" then the order is not specified. If the `ovf:order` is non-zero then the of execution of the
 963 start action shall be the numerical ascending order of the values. The `Items` with same order
 964 identifier may be started concurrently.

965 The order of execution of the stop action should be the numerical descending order of the
 966 values. In implementation specific scenarios the order of execution of the stop action may be
 967 non-descending.

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

- 969 • `ovf:startDelay` specifies a delay in seconds to wait until proceeding to the next virtual
970 system in the start sequence. The default value is 0.
- 971 • `ovf:waitingForGuest` enables the virtualization platform to resume the startup sequence
972 after the guest software has reported it is ready. The interpretation of this is virtualization
973 platform specific. The default value is FALSE.
- 974 • `ovf:startAction` specifies the start action to use. Valid values are `powerOn` and `none`. The
975 default value is `powerOn`.
- 976 • `ovf:stopDelay` specifies a delay in seconds to wait until proceeding to the previous order in
977 the stop sequence. The default value is 0.
- 978 • `ovf:stopAction` specifies the stop action to use. Valid values are `powerOff`,
979 `guestShutdown`, and `none`. The interpretation of `guestShutdown` is virtualization platform
980 specific. The default value is `powerOff`.

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

982 9.8 DeploymentOptionSection

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

987 The `DeploymentOptionSection` specifies an ID, label, and description for each configuration. See
988 D.16 for an example.

989 The `DeploymentOptionSection` has the following semantics:

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

1000 Configurations may be used to control resources for virtual hardware and for virtual machine collections.
1001 The `Item`, `EthernetPortItem`, and `StorageItem` elements in `VirtualHardwareSection`
1002 elements describe resources for `VirtualSystem` entities, while the `Item`, `EthernetPortItem`, and
1003 `StorageItem` elements in `ResourceAllocationSection` elements describe resources for virtual
1004 machine collections. For these two `Item`, `EthernetPortItem`, or `StorageItem` types, the
1005 following additional semantics are defined:

- 1006 • Each `Item`, `EthernetPortItem`, and `StorageItem` has an optional
1007 `ovf:configuration` attribute, containing a list of configurations separated by a single space
1008 character. If not specified, the item shall be selected for any configuration. If specified, the item
1009 shall be selected only if the chosen configuration ID is in the list. A configuration attribute shall
1010 not contain a configuration ID that is not specified in the `DeploymentOptionSection`.

- 1011 • Within a single `VirtualHardwareSection` or `ResourceAllocationSection`, multiple
1012 `Item`, `EthernetPortItem`, and `StorageItem` elements are allowed to refer to the same
1013 `InstanceID`. A single combined `Item`, `EthernetPortItem`, or `StorageItem` for the
1014 given `InstanceID` shall be constructed by picking up the child elements of each `Item`,
1015 `EthernetPortItem`, or `StorageItem` element, with child elements of a former `Item`,
1016 `EthernetPortItem`, or `StorageItem` element in the OVF descriptor not being picked up
1017 if there is a like-named child element in a latter `Item`, `EthernetPortItem`, or
1018 `StorageItem` element. Any attributes specified on child elements of `Item`,
1019 `EthernetPortItem`, or `StorageItem` elements that are not picked up that way, are not
1020 part of the combined `Item`, `EthernetPortItem`, or `StorageItem` element.
- 1021 • All `Item`, `EthernetPortItem`, `StorageItem` elements shall specify `ResourceType`, and
1022 `Item`, `EthernetPortItem`, and `StorageItem` elements with the same `InstanceID` shall
1023 agree on `ResourceType`.

1024 Note that the attributes `ovf:configuration` and `ovf:bound` on `Item` may be used in combination to
1025 provide flexible configuration options.

1026 Configurations can further be used to control default values for properties and whether properties are
1027 user configurable. For `Property` elements inside a `ProductSection`, the following additional semantic
1028 is defined:

- 1029 • It is possible to specify alternative default property values for different configurations in a
1030 `DeploymentOptionSection`. In addition to a `Label` and `Description` element, each
1031 `Property` element may optionally contain `Value` elements. The `Value` element shall have
1032 an `ovf:value` attribute specifying the alternative default and an `ovf:configuration`
1033 attribute specifying the configuration in which this new default value should be used. Multiple
1034 `Value` elements shall not refer to the same configuration.
- 1035 • A `Property` element may optionally have an `ovf:configuration` attribute specifying the
1036 configuration in which this property should be user configurable. The value of
1037 `ovf:userConfigurable` is implicitly set to `FALSE` for all other configurations, in which
1038 case the default value of the property may not be changed during installation.

1039 9.9 OperatingSystemSection

1040 An `OperatingSystemSection` specifies the operating system installed on a virtual machine. See D.17
1041 for an example.

1042 The values for `ovf:id` should be taken from the `ValueMap` of the `CIM_OperatingSystem.OsType`
1043 property. The description should be taken from the corresponding `Values` of the
1044 `CIM_OperatingSystem.OsType` property.

1045 The `OperatingSystemSection` element is a valid section for a `VirtualSystem` element only.

1046 9.10 InstallSection

1047 The `InstallSection` element, if specified, indicates that the virtual machine needs to be booted once
1048 in order to install and/or configure the guest software. The guest software is expected to access the OVF
1049 environment during that boot, and to shut down after having completed the installation and/or
1050 configuration of the software, powering off the guest.

1051 If the `InstallSection` is not specified, this indicates that the virtual machine does not need to be
1052 powered on to complete installation of guest software. See D.18 for an example.

1053 The `InstallSection` element shall be valid only for a `VirtualSystem` element.

1054 The optional `ovf:initialBootStopDelay` attribute specifies a delay in seconds to wait for the virtual
1055 machine to power off. If not set, the implementation shall wait for the virtual machine to power off by itself.
1056 If the delay expires and the virtual machine has not powered off, the consumer of the OVF package shall
1057 indicate a failure.

1058 Note that the guest software in the virtual machine can do multiple reboots before powering off.

1059 Several virtual systems in a virtual system collection may have an `InstallSection` element defined, in
1060 which case the above step is done for each virtual system that has an `InstallSection` element.-.

1061 9.11 EnvironmentFilesSection

1062 The `EnvironmentFilesSection` enables the OVF package to specify additional environment file(s)
1063 (AEF) besides the virtual disks. These AEF's enable increased flexibility in image customization outside of
1064 virtual disk capture, allowing an OVF package to provide customized solutions by combining existing
1065 virtual disks without modifying them.

1066 The AEF contents are neither generated nor validated by the deployment function.

1067 The AEF's are included in the transport media generated by the deployment function.

1068 The AEF's are conveyed to the guest software using the indicated transport media type. The AEF's and
1069 OVF environment files are intended to use same transport media and transport media type

1070 The `EnvironmentFilesSection` shall contain a `File` element with the attributes `ovf:fileRef` and
1071 `ovf:path` for each AEF provided to the guest software.

1072 The `ovf:fileRef` attribute shall specify an existing `File` element in the `References` element. The
1073 `File` element is identified by matching its `ovf:id` attribute value with the `ovf:fileRef` attribute value.

1074 The `ovf:path` attribute specifies the relative location in the transport media (see clause 11.1) where the
1075 file should be placed, using the syntax of relative-path references in [RFC3986](#).

1076 The referenced `File` element in the `References` element identifies the content using one of the URL
1077 schemes "file", "http", or "https". For the "file" scheme, the content is static and included in
1078 the OVF package. See ANNEX F for deployment considerations

1079 For details about transport media type, see clause 11.2.

1080 9.12 BootDeviceSection

1081 Individual virtual machines use the default device boot order provided by the virtualization platform's
1082 virtual BIOS. The `BootDeviceSection` allows the OVF package author to specify particular boot
1083 configurations and boot order settings. This enables booting from non-default devices such as a NIC
1084 using PXE, a USB device or a secondary disk. Moreover there could be multiple boot configurations with
1085 different boot orders. For example, a virtual disk may need to be patched before it is bootable and a patch
1086 ISO image could be included in the OVF package.

1087 The Common Information Model (CIM) defines artifacts to deal with boot order use cases prevalent in the
1088 industry for BIOSes found in desktops and servers. The boot configuration is defined by the class
1089 `CIM_BootConfigSetting` that in turn contains one or more `CIM_BootSourceSetting` classes as
1090 defined in the CIM schema. Each class representing the boot source in turn has either the specific device
1091 or a "device type" such as disk or CD/DVD as a boot source.

1092 In the context of this specification, the `InstanceID` property of `CIM_BootSourceSetting` is used for
1093 identifying a specific device as the boot source. The `InstanceID` property of the device as specified in
1094 the `Item` description of the device in the `VirtualHardwareSection` element is used to specify the

1095 device as a boot source. In case the source is desired to be a device type, the
1096 `StructuredBootString` field is used to denote the type of device with values defined by the CIM boot
1097 control profile. See ANNEX F for deployment considerations.

1098 See D.21 for an example.

1099 9.13 SharedDiskSection

1100 The existing `DiskSection` element in clause 9.1 describes virtual disks in the OVF package. Virtual
1101 disks in the `DiskSection` element can be referenced by multiple virtual systems, but seen from the
1102 guest software each virtual system gets individual private disks. Any level of sharing done at runtime is
1103 deployment platform specific and not visible to the guest software.

1104 Certain applications such as clustered databases rely on multiple virtual systems sharing the same virtual
1105 disk at runtime. `SharedDiskSection` allows the OVF package to specify `Disk` elements shared by
1106 more than one virtual system at runtime, these could be virtual disks backed by an external `File`
1107 reference, or empty virtual disks without backing. It is recommended that the guest software use cluster-
1108 aware file system technology to be able to handle concurrent access. See D.21 for an example.

1109 The `SharedDiskSection` is a valid section at the outermost envelope level only.

1110 Each virtual disk is represented by a `SharedDisk` element that shall be given an identifier using the
1111 `ovf:diskId` attribute; the identifier shall be unique within the combined content of `DiskSection` and
1112 `SharedDiskSection` element. The `SharedDisk` element has the same structure as the `Disk` element
1113 in the `DiskSection` element, with the addition of an `ovf:readOnly` attribute. The `ovf:readOnly` is
1114 optional and states whether shared disk access is read-write i.e. FALSE, or read-only i.e., TRUE.

1115 Shared virtual disks are referenced from virtual hardware using the `HostResource` element as described
1116 in clause 8.3.

1117 It is optional for the virtualization platform to support the `SharedDiskSection` element. The platform
1118 should give an appropriate error message based on the value of the `ovf:required` attribute on the
1119 `SharedDiskSection` element.

1120 9.14 ScaleOutSection

1121 The number of virtual systems or collections of virtual system contained in an OVF package is fixed and
1122 determined by the structure inside the `Envelope` element. The `ScaleOutSection` element allows a
1123 `VirtualSystemCollection` element to contain a set of children that are homogeneous with respect to
1124 a prototypical `VirtualSystem` or `VirtualSystemCollection` element. The `ScaleOutSection`
1125 element shall cause the deployment function to replicate the prototype a number of times, thus allowing
1126 the number of instantiated virtual systems to be configured dynamically at deployment time. See for an
1127 example.

1128 This mechanism enables scaling of virtual system instances at deployment time. Scaling at runtime is not
1129 within the scope of this specification.

1130 The `ScaleOutSection` element is a valid section inside `VirtualSystemCollection` element only.

1131 The `ovf:id` attribute on `ScaleOutSection` element identifies the virtual system or collection of
1132 virtual systems prototype to be replicated.

1133 For the `InstanceCount` element, the `ovf:minimum` and `ovf:maximum` attribute values shall be non-
1134 negative integers and `ovf:minimum` shall be less than or equal to the value of `ovf:maximum`. The
1135 `ovf:minimum` value may be zero in which case the `VirtualSystemCollection` may contain zero instances
1136 of the prototype. If the `ovf:minimum` attribute is not present, it shall be assumed to have a value of one.

1137 If the `ovf:maximum` attribute is not present, it shall be assumed to have a value of unbounded. The
 1138 `ovf:default` attribute is required and shall contain a value within the range defined by `ovf:minimum`
 1139 and `ovf:maximum`.

1140 Each replicated instance shall be assigned a unique `ovf:id` value within the
 1141 `VirtualSystemCollection` element. The unique `ovf:id` value shall be constructed from the
 1142 prototype `ovf:id` value with a sequence number appended as follows:

```
1143 replica-ovf-id = prototype-ovf-id "-" decimal-number
1144 decimal-number = decimal-digit | (decimal-digit decimal-number)
1145 decimal-digit  = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
```

1146 The syntax definitions above use ABNF with the exceptions listed in ANNEX A. The first replica shall
 1147 have sequence number one and following sequence numbers shall be incremented by one for each
 1148 replica. Note that after deployment, no `VirtualSystem` will have the prototype `ovf:id` value itself.

1149 If the prototype being replicated has a starting order in the `StartupSection`, all replicas shall share this
 1150 value. It is not possible to specify a particular starting sequence among replicas.

1151 Property values for Property elements in the prototype are prompted for once per replica created. If the
 1152 OVF package author requires a property value to be shared among instances, that Property may be
 1153 declared at the containing `VirtualSystemCollection` level and referenced by replicas as described in
 1154 clause 9.5.

1155 Configurations from the `DeploymentOptionSection` element may be used to control values for
 1156 `InstanceCount` element. The `InstanceCount` element may have an `ovf:configuration` attribute
 1157 specifying the configuration in which this element should be used. Multiple elements shall not refer to the
 1158 same configuration, and a configuration attribute shall not contain an `ovf:id` value that is not specified in
 1159 the `DeploymentOptionSection`. See D.22 for an example.

1160 9.15 PlacementGroupSection and PlacementSection

1161 Certain types of applications require the ability to specify that two or more `VirtualSystem` elements
 1162 should be deployed closely together since they rely on very fast communication or a common hardware
 1163 dependency such as a reliable communication link. Other types of applications require the ability to
 1164 specify that two or more `VirtualSystems` should be deployed apart due to high-availability or disaster
 1165 recovery considerations.

1166 The `PlacementGroupSection` allows an OVF package to define a placement policy for a group of
 1167 `VirtualSystems`, while the `PlacementSection` allows the annotation of the elements with
 1168 membership of a particular placement policy group.

1169 Zero or more `PlacementGroupSections` may be declared at the `Envelope` level, while the
 1170 `PlacementSection` may be declared at the `VirtualSystem` or `VirtualSystemCollection` level. Declaring a
 1171 `VirtualSystemCollection` member of a placement policy group applies transitively to all child `VirtualSystem`
 1172 and child `Virtual System Collections` elements. The `ovf:id` attribute on `PlacementGroupSection` is
 1173 used to identify the particular placement policy; the attribute value shall be unique within the OVF
 1174 package. Placement policy group membership is specified using the `ovf:group` attribute on
 1175 `PlacementSection`; the attribute value shall match the value of an `ovf:id` attribute on a
 1176 `PlacementGroupSection`.

1177 This standard defines the placement policies "affinity" and "availability", specified with the
 1178 required `ovf:policy` attribute on `PlacementGroupSection`.

1179 Placement policy "affinity" describe that `VirtualSystems` should be placed as closely together as
 1180 possible. The deployment platform should attempt to keep these virtual machines located as adjacently
 1181 as possible, typically on the same physical host or with fast network connectivity between hosts.

1182 Placement policy "availability" describe that VirtualSystems should be placed separately. The
 1183 deployment platform should attempt to keep these virtual machines located apart, typically on the
 1184 different physical hosts. See D.23 for an example.

1185 9.16 Encryption Section

1186 It is desirable for licensing and other reasons to have an encryption scheme enabling free exchange of
 1187 OVF appliances while ensuring that only the intended parties can use them. The XML Encryption Syntax
 1188 and Processing standard is utilized to encrypt either the files in the reference section or any parts of the
 1189 XML markup of an OVF document.

1190 The various aspects of OVF encryption are as shown below:

- 1191 1. block encryption
 1192 The OVF package shall utilize block encryption algorithms as specified in the XML encryption 1.1
 1193 documents (ref) for this purpose.
- 1194 2. key derivation
 1195 The OVF package may use the appropriate key for this purpose. If the key is derived using a
 1196 passphrase then the author shall use one of the key derivations specified in the XML Encryption
 1197 1.1 standard.
- 1198 3. key transport.
 1199 If the encryption key is embedded in the OVF package, the specified key transport mechanisms
 1200 shall be used.

1201 This standard defines a section called the EncryptionSection as a focal point for the encryption
 1202 functionality. This section provides a single location for placing the encryption algorithm related markup
 1203 and the corresponding reference list to point to the OVF content that has been encrypted.

1204 Note that depending on the parts of the OVF package that has been encrypted, an OVF descriptor may
 1205 not validate against the DSP8023 until decrypted. See D.24 for an example.

1206 10 Internationalization

1207 The following elements support localizable messages using the optional `ovf:msgid` attribute:

- 1208 • Info element on Content
- 1209 • Name element on Content
- 1210 • Info element on Section
- 1211 • Annotation element on AnnotationSection
- 1212 • License element on EulaSection
- 1213 • Description element on NetworkSection
- 1214 • Description element on OperatingSystemSection
- 1215 • Description, Product, Vendor, Label, and Category elements on ProductSection
- 1216 • Description and Label elements on Property
- 1217 • Description and Label elements on DeploymentOptionSection
- 1218 • ElementName, Caption and Description subelements on the System element in
 1219 VirtualHardwareSection
- 1220 • ElementName, Caption and Description subelements on Item elements in
 1221 VirtualHardwareSection

- 1222 • `ElementName`, `Caption` and `Description` subelements on `Item` elements in
1223 `ResourceAllocationSection`

1224 The `ovf:msgid` attribute contains an identifier that refers to a message that may have different values in
1225 different locales. See D.25 for an example.

1226 The `xml:lang` attribute on the `Envelope` element shall specify the default locale for messages in the
1227 descriptor. The attribute is optional with a default value of "en-US".

1228 10.1 Internal Resource Bundles

1229 Message resource bundles can be internal or external to the OVF descriptor. Internal resource bundles
1230 are represented as `Strings` elements at the end of the `Envelope` element. See D.25 for an example.

1231 10.2 External Resource Bundles

1232 External resource bundles shall be listed first in the `References` section and referred to from `Strings`
1233 elements. An external message bundle follows the same schema as the embedded one. Exactly one
1234 `Strings` element shall be present in an external message bundle, and that `Strings` element shall not
1235 have an `ovf:fileRef` attribute specified. See D.25 for an example.

1236 The embedded and external `Strings` elements may be interleaved, but they shall be placed at the end
1237 of the `Envelope` element. If multiple occurrences of a `msgid` attribute with a given locale occur, a latter
1238 value overwrites a former.

1239 10.3 Message Content in External File

1240 The content of all localizable messages may be stored in an external file using the optional
1241 `ovf:fileRef` attribute on the `Msg` element. For the `License` element on `EulaSection` in particular,
1242 this allows inclusion of a standard license text file in unaltered form without any XML header or footer.

1243 The `ovf:fileRef` attribute denotes the message content by identifying an existing `File` element in the
1244 `References` element, the `File` element is identified by matching its `ovf:id` attribute value with the
1245 `ovf:fileRef` attribute value. The content of an external file referenced using `ovf:fileRef` shall be
1246 interpreted as plain text in UTF-8 Unicode.

1247 If the referenced file is not available, the embedded content of the `Msg` element shall be used.

1248 The optional `ovf:fileRef` attribute may appear on `Msg` elements in both internal and external `Strings`
1249 resource bundles. See D.26 for an example.

1250 11 OVF Environment and OVF Environment File

1251 The OVF environment defines how the guest software and the virtualization platform interact. The OVF
1252 environment enables the guest software to access information about the virtualization platform, such as
1253 the user-specified values for the properties defined in the OVF descriptor.

1254 The `dsp8027_1.1.0.xsd` XML schema definition file for the OVF environment contains the elements
1255 and attributes that define the format and semantics of an XML document which constituent OVF
1256 environment file (OEF). The OEF shall validate with the `DSP8027_1.1.0.xsd`

1257 The OEF is created on a per virtual system basis by the deployment function. The basis of the OEF is the
1258 OVF descriptor, OVF operational metadata, OVF property values, policy metadata, and other user
1259 provided values.

- 1260 OEF is provided to the guest software about the environment in which it is being executed. The way that
1261 the OEF is obtained depends on the transport media type. See D.27 for an example.
- 1262 The value of the `ovfenv:id` attribute of the `Environment` element shall match the value of the `ovf:id`
1263 attribute of the `VirtualSystem` entity describing this virtual system.
- 1264 The `PlatformSection` element contains optional information provided by the deployment function. The
1265 `Kind`, `Version`, and `Vendor` elements describe the virtualization platform.. The `Locale` and `TimeZone`
1266 elements describe the current locale and time zone.
- 1267 The `PropertySection` element contains `Property` elements with key/value pairs corresponding to all
1268 properties specified in the OVF descriptor for the current virtual machine, as well as properties specified
1269 for the immediate parent `VirtualSystemCollection`, if one exists. The environment presents
1270 properties as a simple list to make it easy for applications to parse. Furthermore, the single list format
1271 supports the override semantics where a property on a `VirtualSystem` may override one defined on a
1272 parent `VirtualSystemCollection`. The overridden property shall not be in the list. Overriding may
1273 occur if a property in the current virtual machine and a property in the parent
1274 `VirtualSystemCollection` has identical `ovf:key`, `ovf:class`, and `ovf:instance` attribute
1275 values; see 9.5. In this case, the value of an overridden parent property may be obtained by adding a
1276 differently named child property referencing the parent property with a macro; see 9.5.
- 1277 An `Entity` element shall exist for each sibling `VirtualSystem` and `VirtualSystemCollection`, if
1278 any are present. The value of the `ovfenv:id` attribute of the `Entity` element shall match the value of
1279 the `ovf:id` attribute of the sibling entity. The `Entity` elements contain the property key/value pairs in
1280 the sibling's OVF environment documents, so the content of an `Entity` element for a particular sibling
1281 shall contain the exact `PropertySection` seen by that sibling. This information can be used, for
1282 example, to make configuration information such as IP addresses available to `VirtualSystems` being
1283 part of a multi-tiered application.
- 1284 Table 8 shows the core sections that are defined.

1285 **Table 8 – Core Sections**

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

- 1286 The e is extensible by providing new section types. The deployment function should ignore unknown
1287 section types and elements specified in OEF.

1288 11.1 Transport Media

- 1289 The transport media refers to the format in which information is conveyed to the guest software. The
1290 transport media (e.g. ISO image) is generated by the deployment function that includes OVF environment
1291 file and any additional environment file(s).
- 1292 If the transport media type is 'iso' the generated ISO image shall comply with the ISO 9660 specification
1293 with support for Joliet extensions.
- 1294 The transport media shall contain the OVF environment file for this particular virtual machine shall be
1295 presented in an XML file named `ovf-env.xml` that is contained in the root directory of the transport
1296 media. The guest software can now access the information.

1297 For additional environment file (s), the transport media shall have the root location relative to the
1298 `ovf:path` attribute shall be in a directory named "ovffiles" contained in the root directory . This provides
1299 an access mechanism for the guest software.

1300 Other custom transport media may support this mechanism. Custom transport medium shall specify how
1301 to access multiple data sources from a root location. See D.19\ for an example. The access mechanism
1302 for the guest software is not specified.

1303 11.2 Transport Media Type

1304 The transport media type refers to a mechanism to convey transport media over any data link or
1305 removable storage medium (i.e. CD/DVD-ROM) from deployment functions to guest software.

1306 The `iso` transport media type shall support this mechanism, see clause 11.2 for details.

1307 This standard defines the "iso" transport type to meet the need for interoperability.

1308 The transport media can be communicated in a number of ways to the guest software. These ways are
1309 called transport media types. The transport media types are specified in the OVF descriptor by the
1310 `ovf:transport` attribute of `VirtualHardwareSection`. Several transport media types may be
1311 specified, separated by a single space character, in which case an implementation is free to use any of
1312 them.

1313 To enable interoperability, this specification defines an "iso" transport media type which all
1314 implementations that support CD-ROM devices are required to support. The `iso` transport media type
1315 communicates the environment document by making a dynamically generated ISO image available to the
1316 guest software.

1317 To support the `iso` transport media type, prior to booting a virtual system, an implementation shall make
1318 an ISO read-only disk image available as backing for a disconnected CD-ROM. If the `iso` transport
1319 media type is selected for a `VirtualHardwareSection`, at least one disconnected CD-ROM device
1320 shall be present in this section.

1321 If the virtual system prior to booting had more than one disconnected CD-ROM, the guest software may
1322 have to scan connected CD-ROM devices in order to locate the ISO image containing the `ovf-env.xml`
1323 file.

1324 The transport media containing the OVF environment file shall be made available to the guest software
1325 on every boot of the virtual system.

1326 Support for the "iso" transport media type is not a requirement for virtual hardware architectures or
1327 guest software which do not have CD-ROM device support.

1328 To be conformant with this specification, any transport media type other than `iso` shall be given by a URI
1329 that identifies an unencumbered specification on how to use the transport media type. The specification
1330 need not be machine readable, but it shall be static and unique so that it may be used as a key by
1331 software reading an OVF descriptor to uniquely determine the transport media type. The specification
1332 shall be sufficient for a skilled person to properly interpret the transport media type mechanism for
1333 implementing the protocols. The URIs should be resolvable.

1334

ANNEX A (informative)

1335
1336
1337
1338

Symbols and Conventions

1339 XML examples use the XML namespace prefixes defined in Table 1. The XML examples use a style to
1340 not specify namespace prefixes on child elements. Note that XML rules define that child elements
1341 specified without namespace prefix are from the namespace of the parent element, and not from the
1342 default namespace of the XML document. Throughout the document, whitespace within XML element
1343 values is used for readability. In practice, a service can accept and strip leading and trailing whitespace
1344 within element values as if whitespace had not been used.

1345 Syntax definitions in this document use Augmented BNF (ABNF) as defined in IETF [RFC5234](#) with the
1346 following exceptions:

- 1347 • Rules separated by a bar (|) represent choices, instead of using a forward slash (/) as defined in
1348 ABNF.
- 1349 • Any characters must be processed case sensitively, instead of case-insensitively as defined in
1350 ABNF.
- 1351 • Whitespace (i.e., the space character U+0020 and the tab character U+0009) is allowed between
1352 syntactical elements, instead of assembling elements without whitespace as defined in ABNF.

1353

ANNEX B (normative)

OVF XSD

1354
1355
1356
1357

1358 Normative copies of the XML schemas for this specification may be retrieved by resolving the following
1359 URLs:

1360

1361 <http://schemas.dmtf.org/ovf/envelope/2/dsp8023.xsd>

1362 <http://schemas.dmtf.org/ovf/environment/1/dsp8027.xsd>

1363 Any `xs:documentation` content in XML schemas for this specification is informative and provided only
1364 for convenience.

1365 Normative copies of the XML schemas for the WS-CIM mapping ([DSP0230](#)) of
1366 `CIM_ResourceAllocationSystemSettingsData`, `CIM_VirtualSystemSettingData`,
1367 `CIM_EthernetPortAllocationSettingData`, `CIM_StorageAllocationSettingData` and
1368 `CIM_OperatingSystem`, may be retrieved by resolving the following URLs:

1369

1370 http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData.xsd

1371 <http://schemas.dmtf.org/wbem/wscim/1/cim->

1372 [schema/2/CIM_ResourceAllocationSettingData.xsd](http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData.xsd)

1373 <http://schemas.dmtf.org/wbem/wscim/1/cim->

1374 [schema/2/CIM_EthernetPortAllocationSettingData.xsd](http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_EthernetPortAllocationSettingData.xsd)

1375 http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_StorageAllocationSettingData.xsd

1376 This specification is based on the following CIM MOFs:

1377 `CIM_VirtualSystemSettingData.mof`

1378 `CIM_ResourceAllocationSettingData.mof`

1379 `CIM_EthernetPortAllocationSettingData.mof`

1380 `CIM_StorageAllocationSettingData.mof`

1381 `CIM_OperatingSystem.mof`

1382

ANNEX C (informative)

OVF Mime Type Registration Template

- 1383
1384
1385
1386
- 1387 Registration Template
- 1388 To: ietf-types@iana.org
- 1389 Subject: Registration of media type Application/OVF
- 1390 Type name: Application
- 1391 Subtype name: OVF
- 1392 Required parameters: none
- 1393 Optional parameters: none
- 1394 Encoding considerations: binary
- 1395 Security considerations:
- 1396 • An OVF package contains active content that is expected to be launched in a virtual machine.
1397 The OVF standard, section 5.1 states: “An OVF package may be signed by signing the manifest
1398 file. The digest of the manifest file is stored in a certificate file with extension .cert file along with
1399 the base64-encoded X.509 certificate. The .cert file shall have the same base name as the .ovf
1400 file and be a sibling of the .ovf file. A consumer of the OVF package shall verify the signature and
1401 should validate the certificate.
 - 1402 • An OVF package may contain passwords as part of the configuration information. The OVF
1403 standard, section 9.5 states: “The optional Boolean attribute ovf:password indicates that the
1404 property value may contain sensitive information. The default value is FALSE. OVF
1405 implementations prompting for property values are advised to obscure these values when
1406 ovf:password is set to TRUE. This is similar to HTML text input of type password. Note that this
1407 mechanism affords limited security protection only. Although sensitive values are masked from
1408 casual observers, default values in the OVF descriptor and assigned values in the OVF
1409 environment are still passed in clear text. “
- 1410 Interoperability considerations:
- 1411 • OVF has demonstrated interoperability via multiple, interoperating implementations in the market.
- 1412 Published specification:
- 1413 • DSP0243_2.0.0.pdf
- 1414 Applications that use this media type:
- 1415 • Implementations of the DMTF Standard: Cloud Infrastructure Management Interface (CIMI)
1416 (DSP0263_1.0.0.pdf)
 - 1417 • Implementations of the SNIA Cloud Data Management Interface (CDMI) – OVF Extension
- 1418 Additional information:
- 1419 • Magic number(s): none
 - 1420 • File extension(s): .ova
 - 1421 • Macintosh file type code(s): none
 - 1422 • Person & email address to contact for further information:

- 1423 • Intended usage: (One of COMMON, LIMITED USE or OBSOLETE.)
- 1424 • Restrictions on usage: (Any restrictions on where the media type can be used go here.)
- 1425 • Author:
- 1426 • Change controller:
- 1427

ANNEX D (informative)

OVF Examples

1428
1429
1430
1431

D.1 Examples of OVF Package Structure

1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442

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
```

1443
1444
1445
1446

EXAMPLE 2:

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

```
SHA256(package.ovf)= 9902cc5ec4f4a00cabbff7b60d039263587ab430d5fbd5c5e8707391c90a4
SHA256(vmdisk.vmdk)= aab66c4d70e17cec2236a651a3fc618cafc5ec6424122904dc0b9c286fce40c2
```

1447

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
```

1456

1457

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
```

1462

-----BEGIN CERTIFICATE-----

1463

```
MIIBgjCCASwCAQQwDQYJKoZIhvcNAQEEBQAwwODELMAkGA1UEBhMCQVUxDDAKBgNV
```

1464

```
BAGTA1FMRDEbMBkGA1UEAxMSU1NMZWF5L3JzYSB0ZXN0IENBMB4XD Tk1MTAwOTIz
```

1465

```
MzIwNVVoXDTk4MDCwNTIzMzIwNVVowYDELMAkGA1UEBhMCQVUxDDAKBgNVBAGTA1FM
```

1466

```
RDEZMbcGA1UEChMQTWluY29tIFB0eS4gTHRkLjELMAkGA1UECXMxMCQ1MxGzAZBgNV
```

1467

```
BAMTElNTTGvheSBkZW1vIHNLcnZlcjBcMA0GCSqGSIb3DQEBAQUAA0sAMEgCQQC3
```

1468

```
LCXcScWua0PFLkHBLm2VejqpA1F4RQ8q0VjRiPafjx/Z/aWH3ipdMVvuJGa/wFXb
```

1469

```
/nDFLDlFwP+oCPwhBtVPAgMBAAEwDQYJKoZIhvcNAQEEBQADQQArNFsihWIjBzb0
```

1470

```
DcsU0BvL2bvSwJrPEqFlkDq3F4M6EgutL9axEcANWgbbEdAvNJD1dmEmoWny27Pn
```

1471

```
Ims6ZOZB
```

1472

-----END CERTIFICATE-----

D.2 Examples of Distribution of Files

1473

EXAMPLE 1:

1474

An example of an OVF package as a compressed archive:

1475

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

1476

1477

1478 EXAMPLE 2:
 1479 An example of an OVF package as a set of files on Web server follows:
 1480 <http://mywebsite/virtualappliances/package.ovf>
 1481 <http://mywebsite/virtualappliances/vmdisk1.vmdk>
 1482 <http://mywebsite/virtualappliances/vmdisk2.vmdk>
 1483 <http://mywebsite/virtualappliances/resource.iso>
 1484 <http://mywebsite/virtualappliances/de-DE-resources.xml>

1485 D.3 Example of Envelope Element

1486 An example of the structure of an OVF descriptor with the top-level Envelope element
 1487 follows:

```

1488 <?xml version="1.0" encoding="UTF-8"?>
1489 <Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1490   xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-
1491   schema/2/CIM_VirtualSystemSettingData"
1492   xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
1493   schema/2/CIM_ResourceAllocationSettingData"
1494   xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/2"
1495   xmlns="http://schemas.dmtf.org/ovf/envelope/2"
1496   xml:lang="en-US">
1497   <References>
1498     <File ovf:id="de-DE-resources.xml" ovf:size="15240"
1499       ovf:href="http://mywebsite/virtualappliances/de-DE-resources.xml"/>
1500     <File ovf:id="file1" ovf:href="vmdisk1.vmdk" ovf:size="180114671"/>
1501     <File ovf:id="file2" ovf:href="vmdisk2.vmdk" ovf:size="4882023564"
1502   ovf:chunkSize="2147483648"/>
1503     <File ovf:id="file3" ovf:href="resource.iso" ovf:size="212148764"
1504   ovf:compression="gzip"/>
1505     <File ovf:id="icon" ovf:href="icon.png" ovf:size="1360"/>
1506   </References>
1507   <!-- Describes meta-information about all virtual disks in the package -->
1508   <DiskSection>
1509     <Info>Describes the set of virtual disks</Info>
1510     <!-- Additional section content -->
1511   </DiskSection>
1512   <!-- Describes all networks used in the package -->
1513   <NetworkSection>
1514     <Info>List of logical networks used in the package</Info>
1515     <!-- Additional section content -->
1516   </NetworkSection>
1517   <SomeSection ovf:required="false">
1518     <Info>A plain-text description of the content</Info>
1519     <!-- Additional section content -->
1520   </SomeSection>
1521   <!-- Additional sections can follow -->
1522   <VirtualSystemCollection ovf:id="Some Product">
1523     <!-- Additional sections including VirtualSystem or VirtualSystemCollection-->
1524   </VirtualSystemCollection >
1525   <Strings xml:lang="de-DE">
1526     <!-- Specification of message resource bundles for de-DE locale -->
1527   </Strings>
1528 </Envelope>

```

1529 D.4 Example of File References

1530 EXAMPLE 1:

1531 The following example shows different types of file references:

```
1532 <File ovf:id="disk1" ovf:href="disk1.vmdk"/>
1533 <File ovf:id="disk2" ovf:href="disk2.vmdk" ovf:size="5368709120"
1534         ovf:chunkSize="2147483648"/>
1535 <File ovf:id="iso1" ovf:href="resources/image1.iso"/>
1536 <File ovf:id="iso2" ovf:href="http://mywebsite/resources/image2.iso"/>
1537
```

1538 EXAMPLE 2:

1539 The following example shows manifest entries corresponding to the file references
1540 above:

```
1541 SHA1(disk1.vmdk)= 3e19644ec2e806f38951789c76f43e4a0ec7e233
1542 SHA1(disk2.vmdk.000000000)= 4f7158731fff434380bf217da248d47a2478e79d8
1543 SHA1(disk2.vmdk.000000001)= 12849daeeaf43e7a89550384d26bd437bb8defaf
1544 SHA1(disk2.vmdk.000000002)= 4cdd21424bd9eeafa4c42112876217de2ee5556d
1545 SHA1(resources/image1.iso)= 72b37ff3fdd09f2a93f1b8395654649b6d06b5b3
1546 SHA1(http://mywebsite/resources/image2.iso)=
1547 d3c2d179011c970615c5cf10b30957d1c4c968ad
```

1548 D.5 Example of Content Element

1549 An example of a VirtualSystem element structure follows:

```
1550 <VirtualSystem ovf:id="simple-app">
1551   <Info>A virtual machine</Info>
1552   <Name>Simple Appliance</Name>
1553   <SomeSection>
1554     <!-- Additional section content -->
1555   </SomeSection>
1556   <!-- Additional sections can follow -->
1557 </VirtualSystem>
1558
```

1559 An example of a VirtualSystemCollection element structure follows:

```
1560 <VirtualSystemCollection ovf:id="multi-tier-app">
1561   <Info>A collection of virtual machines</Info>
1562   <Name>Multi-tiered Appliance</Name>
1563   <SomeSection>
1564     <!-- Additional section content -->
1565   </SomeSection>
1566   <!-- Additional sections can follow -->
1567   <VirtualSystem ovf:id="...">
1568     <!-- Additional sections -->
1569   </VirtualSystem>
1570   <!-- Additional VirtualSystem or VirtualSystemCollection elements can follow-->
1571 </VirtualSystemCollection>
```

1572 D.6 Examples of Extensibility

1573 EXAMPLE 1:

```
1574 <!-- Optional custom section example -->
1575 <othersns:IncidentTrackingSection ovf:required="false">
1576   <Info>Specifies information useful for incident tracking purposes</Info>
1577   <BuildSystem>Acme Corporation Official Build System</BuildSystem>
1578   <BuildNumber>102876</BuildNumber>
1579   <BuildDate>10-10-2008</BuildDate>
1580 </othersns:IncidentTrackingSection>
```

```

1581
1582 EXAMPLE 2:
1583     <!-- Open content example (extension of existing type) -->
1584     <AnnotationSection>
1585         <Info>Specifies an annotation for this virtual machine</Info>
1586         <Annotation>This is an example of how a future element (Author) can still be
1587             parsed by older clients</Annotation>
1588         <!-- AnnotationSection extended with Author element -->
1589         <others:Author ovf:required="false">John Smith</others:Author>
1590     </AnnotationSection>
1591
1592 EXAMPLE 3:
1593     <!-- Optional custom attribute example -->
1594     <Network ovf:name="VM network" others:desiredCapacity="1 Gbit/s">
1595         <Description>The main network for VMs</Description>
1596     </Network>

```

1597 D.7 Examples of VirtualHardwareSection

```

1598 EXAMPLE 1:
1599 Example of VirtualHardwareSection:
1600 <VirtualHardwareSection>
1601     <Info>Memory = 4 GB, CPU = 1 GHz, Disk = 100 GB, 1 Ethernet nic</Info>
1602     <Item>
1603         <rasd:AllocationUnits>Hertz*10^9</rasd:AllocationUnits>
1604         <rasd:Description>Virtual CPU</rasd:Description>
1605         <rasd:ElementName>1 GHz virtual CPU</rasd:ElementName>
1606         <rasd:InstanceID>1</rasd:InstanceID>
1607         <rasd:Reservation>1</rasd:Reservation>
1608         <rasd:ResourceType>3</rasd:ResourceType>
1609         <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
1610         <rasd:VirtualQuantityUnit>Count</ rasd:VirtualQuantityUnit>
1611     </Item>
1612     <Item>
1613         <rasd:AllocationUnits>byte*2^30</rasd:AllocationUnits>
1614         <rasd:Description>Memory</rasd:Description>
1615         <rasd:ElementName>1 GByte of memory</rasd:ElementName>
1616         <rasd:InstanceID>2</rasd:InstanceID>
1617         <rasd:Limit>4</rasd:Limit>
1618         <rasd:Reservation>4</rasd:Reservation>
1619         <rasd:ResourceType>4</rasd:ResourceType>
1620     </Item>
1621     <EthernetPortItem>
1622         <rasd:AllocationUnits>bit / second *2^30 </rasd:AllocationUnits>
1623         <epasd:Connection>VM Network</epasd:Connection>
1624         <epasd:Description>Virtual NIC</epasd:Description>
1625         <epasd:ElementName>Ethernet Port</epasd:ElementName>
1626         <epasd:NetworkPortProfileID>1</epasd:NetworkPortProfileID>
1627         <epasd:NetworkPortProfileIDType>4</epasd:NetworkPortProfileIDType>
1628         <epasd:ResourceType>10</epasd:ResourceType>
1629         <epasd:VirtualQuantity>1</epasd:VirtualQuantity>
1630         <epasd:VirtualQuantityUnits>Count</epasd:VirtualQuantityUnits>
1631     </EthernetPortItem>
1632     <StorageItem>
1633         <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
1634         <sasd:Description>Virtual Disk</sasd:Description>

```

```

1635     <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
1636     <sasd:Reservation>100</sasd:Reservation>
1637     <sasd:ResourceType>31</sasd:ResourceType>
1638     <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
1639     <sasd:VirtualQuantityUnit>Count</sasd:VirtualQuantityUnit>
1640   </StorageItem>
1641 </VirtualHardwareSection>
1642
1643 EXAMPLE 2:
1644     <rasd:ResourceSubType>buslogic lsilogic</rasd:ResourceSubType>

```

1645 D.8 Examples of Virtual Hardware Elements

1646 EXAMPLE 1:
 1647 The following example shows a description of memory size:

```

1648   <Item>
1649     <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1650     <rasd:Description>Memory Size</rasd:Description>
1651     <rasd:ElementName>256 MB of memory</rasd:ElementName>
1652     <rasd:InstanceID>2</rasd:InstanceID>
1653     <rasd:ResourceType>4</rasd:ResourceType>
1654     <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
1655   </Item>

```

1657 EXAMPLE 2:
 1658 The following example shows a description of a virtual Ethernet adapter:

```

1659   <EthernetPortItem>
1660     <epasd:Address>00-16-8B-DB-00-5E</epasd:Address>
1661     <epasd:Connection>VM Network</epasd:Connection>
1662     <epasd:Description>Virtual NIC</epasd:Description>
1663
1664     <epasd:ElementName>Ethernet Port 1</epasd:ElementName>
1665     <epasd:InstanceID>3</epasd:InstanceID>
1666     <epasd:NetworkPortProfileID>1</epasd:NetworkPortProfileID>
1667     <epasd:NetworkPortProfileIDType>4</epasd:NetworkPortProfileIDType>
1668     <epasd:VirtualQuantityUnits>1</epasd:VirtualQuantityUnits>
1669   </EthernetPortItem>

```

1671 EXAMPLE 3:
 1672 The following example shows a description of a virtual storage:

```

1673   <StorageItem>
1674     <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
1675     <sasd:Description>Virtual Disk</sasd:Description>
1676     <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
1677     <sasd:InstanceID>4</sasd:InstanceID>
1678     <sasd:Reservation>100</sasd:Reservation>
1679     <sasd:ResourceType>31</sasd:ResourceType>
1680     <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
1681   </StorageItem>

```

1682 D.9 Example of Ranges on Elements

1683 EXAMPLE:
 1684 The following example shows the use of range markers:
 1685 <VirtualHardwareSection>
 1686 <Info>...</Info>

```

1687 <Item>
1688   <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1689   <rasd:ElementName>512 MB memory size</rasd:ElementName>
1690   <rasd:InstanceID>0</rasd:InstanceID>
1691   <rasd:ResourceType>4</rasd:ResourceType>
1692   <rasd:VirtualQuantity>512</rasd:VirtualQuantity>
1693 </Item>
1694 <Item ovf:bound="min">
1695   <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1696   <rasd:ElementName>384 MB minimum memory size</rasd:ElementName>
1697   <rasd:InstanceID>0</rasd:InstanceID>
1698   <rasd:Reservation>384</rasd:Reservation>
1699   <rasd:ResourceType>4</rasd:ResourceType>
1700 </Item>
1701 <Item ovf:bound="max">
1702   <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1703   <rasd:ElementName>1024 MB maximum memory size</rasd:ElementName>
1704   <rasd:InstanceID>0</rasd:InstanceID>
1705   <rasd:Reservation>1024</rasd:Reservation>
1706   <rasd:ResourceType>4</rasd:ResourceType>
1707 </Item>
1708 </VirtualHardwareSection>

```

1709 D.10 Example of DiskSection

```

1710 EXAMPLE: The following example shows a description of virtual disks:
1711 <DiskSection>
1712   <Info>Describes the set of virtual disks</Info>
1713   <Disk ovf:diskId="vmdisk1" ovf:fileRef="file1" ovf:capacity="8589934592"
1714     ovf:populatedSize="3549324972"
1715     ovf:format=
1716       "http://www.vmware.com/interfaces/specifications/vmdk.html#sparse">
1717   </Disk>
1718   <Disk ovf:diskId="vmdisk2" ovf:capacity="536870912">
1719   </Disk>
1720   <Disk ovf:diskId="vmdisk3" ovf:capacity="${disk.size}"
1721     ovf:capacityAllocationUnits="byte * 2^30">
1722   </Disk>
1723 </DiskSection>

```

1724 D.11 Example of NetworkSection

```

1725 <NetworkSection>
1726   <Info>List of logical networks used in the package</Info>
1727   <Network ovf:name="VM Network">
1728     <Description>The network that the service will be available on</Description>
1729     <NetworkPortProfile>
1730       <Item>
1731         <epasd:AllocationUnits>GigaBits per Second</epasd:AllocationUnits>
1732         <epasd:ElementName>Network Port Profile 1</epasd:ElementName>
1733         <epasd:InstanceID>1</epasd:InstanceID>
1734         <epasd:NetworkPortProfileID>1</epasd:NetworkPortProfileID>
1735         <epasd:NetworkPortProfileIDType>4</epasd:NetworkPortProfileIDType>
1736         <epasd:Reservation>1</epasd:Reservation>
1737       </Item>
1738     </NetworkPortProfile>

```

```
1739     </Network>
1740 </NetworkSection>
```

1741 D.12 Example of ResourceAllocationSection

```
1742 <ResourceAllocationSection>
1743     <Info>Defines reservations for CPU and memory for the collection of VMs</Info>
1744     <Item>
1745         <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1746         <rasd:ElementName>300 MB reservation</rasd:ElementName>
1747         <rasd:InstanceID>0</rasd:InstanceID>
1748         <rasd:Reservation>300</rasd:Reservation>
1749         <rasd:ResourceType>4</rasd:ResourceType>
1750     </Item>
1751     <Item ovf:configuration="..." ovf:bound="...">
1752         <rasd:AllocationUnits>hertz * 10^6</rasd:AllocationUnits>
1753         <rasd:ElementName>500 MHz reservation</rasd:ElementName>
1754         <rasd:InstanceID>0</rasd:InstanceID>
1755         <rasd:Reservation>500</rasd:Reservation>
1756         <rasd:ResourceType>3</rasd:ResourceType>
1757     </Item>
1758     <EthernetPortItem>
1759         <epasd:Address>00-16-8B-DB-00-5E</epasd:Address>
1760         <epasd:Connection>VM Network</epasd:Connection>
1761         <epasd:Description>Virtual NIC</epasd:Description>
1762         <epasd:ElementName>Ethernet Port 1</epasd:ElementName>
1763         <epasd:InstanceID>3</epasd:InstanceID>
1764         <epasd:NetworkPortProfileID>1</epasd:NetworkPortProfileID>
1765         <epasd:NetworkPortProfileIDType>4</epasd:NetworkPortProfileIDType>
1766         <epasd:VirtualQuantityUnits>1</epasd:VirtualQuantityUnits>
1767     </EthernetPortItem>
1768     <StorageItem>
1769         <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
1770         <sasd:Description>Virtual Disk</sasd:Description>
1771         <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
1772         <sasd:InstanceID>4</sasd:InstanceID>
1773         <sasd:Reservation>100</sasd:Reservation>
1774         <sasd:ResourceType>31</sasd:ResourceType>
1775         <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
1776     </StorageItem>
1777 </ResourceAllocationSection>
```

1778 D.13 Example of Annotation

```
1779 <AnnotationSection>
1780     <Info>An annotation on this service. It can be ignored</Info>
1781     <Annotation>Contact customer support if you have any problems</Annotation>
1782 </AnnotationSection >
```

1783 D.14 Example of Product Section

```
1784 <ProductSection ovf:class="com.mycrm.myservice" ovf:instance="1">
1785     <Info>Describes product information for the service</Info>
1786     <Product>MyCRM Enterprise</Product>
1787     <Vendor>MyCRM Corporation</Vendor>
1788     <Version>4.5</Version>
1789     <FullVersion>4.5-b4523</FullVersion>
```

```

1790     <ProductUrl>http://www.mycrm.com/enterprise</ProductUrl>
1791     <VendorUrl>http://www.mycrm.com</VendorUrl>
1792     <Icon ovf:height="32" ovf:width="32" ovf:mimeType="image/png" ovf:fileRef="icon">
1793     <Category>Email properties</Category>
1794     <Property ovf:key="admin.email" ovf:type="string" ovf:userConfigurable="true">
1795         <Label>Admin email</Label>
1796         <Description>Email address of administrator</Description>
1797     </Property>
1798     <Category>Admin properties</Category>
1799     <Property ovf:key="app_log" ovf:type="string" ovf:value="low"
1800 ovf:userConfigurable="true">
1801         <Description>Loglevel for the service</Description>
1802     </Property>
1803     <Property ovf:key="app_isSecondary" ovf:value="false" ovf:type="boolean">
1804         <Description>Cluster setup for application server</Description>
1805     </Property>
1806     <Property ovf:key="app_ip" ovf:type="string" ovf:value="{appserver-vm}">
1807         <Description>IP address of the application server VM</Description>
1808     </Property>
1809 </ProductSection>

```

1810 D.15 Example of EULA Section

```

1811 <EulaSection>
1812     <Info>Licensing agreement</Info>
1813     <License>
1814 Lorem ipsum dolor sit amet, ligula suspendisse nulla pretium, rhoncus tempor placerat
1815 fermentum, enim integer ad vestibulum volutpat. Nisl rhoncus turpis est, vel elit,
1816 congue wisi enim nunc ultricies sit, magna tincidunt. Maecenas aliquam maecenas ligula
1817 nostra, accumsan taciti. Sociis mauris in integer, a dolor netus non dui aliquet,
1818 sagittis felis sodales, dolor sociis mauris, vel eu libero cras. Interdum at. Eget
1819 habitasse elementum est, ipsum purus pede porttitor class, ut adipiscing, aliquet sed
1820 auctor, imperdiet arcu per diam dapibus libero duis. Enim eros in vel, volutpat nec
1821 pellentesque leo, scelerisque.
1822     </License>
1823 </EulaSection>

```

1824 A.1 Example of StartupSection

```

1825 <StartupSection>
1826     <Item ovf:id="vm1" ovf:order="0" ovf:startDelay="30" ovf:stopDelay="0"
1827         ovf:startAction="powerOn" ovf:waitingForGuest="true"
1828 ovf:stopAction="powerOff"/>
1829     <Item ovf:id="teamA" ovf:order="0"/>
1830     <Item ovf:id="vm2" ovf:order="1" ovf:startDelay="0" ovf:stopDelay="20"
1831         ovf:startAction="powerOn" ovf:stopAction="guestShutdown"/>
1832 </StartupSection>

```

1833 D.16 Example of DeploymentOptionSection

```

1834 <DeploymentOptionSection>
1835     <Configuration ovf:id="minimal">
1836         <Label>Minimal</Label>
1837         <Description>Some description</Description>
1838     </Configuration>
1839     <Configuration ovf:id="normal" ovf:default="true">
1840         <Label>Typical</Label>

```

```

1841         <Description>Some description</Description>
1842     </Configuration>
1843     <!-- Additional configurations -->
1844 </DeploymentOptionSection>
1845
1846 EXAMPLE 1: The following example shows a VirtualHardwareSection:
1847 <VirtualHardwareSection>
1848     <Info>...</Info>
1849     <Item>
1850         <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1851         <rasd:ElementName>512 MB memory size and 256 MB reservation</rasd:ElementName>
1852         <rasd:InstanceID>0</rasd:InstanceID>
1853         <rasd:Reservation>256</rasd:Reservation>
1854         <rasd:ResourceType>4</rasd:ResourceType>
1855         <rasd:VirtualQuantity>512</rasd:VirtualQuantity>
1856     </Item>
1857     ...
1858     <Item ovf:configuration="big">
1859         <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1860         <rasd:ElementName>1024 MB memory size and 512 MB reservation</rasd:ElementName>
1861         <rasd:InstanceID>0</rasd:InstanceID>
1862         <rasd:Reservation>512</rasd:Reservation>
1863         <rasd:ResourceType>4</rasd:ResourceType>
1864         <rasd:VirtualQuantity>1024</rasd:VirtualQuantity>
1865     </Item>
1866 </VirtualHardwareSection>

```

```

1867
1868 EXAMPLE 2: The following shows an example ProductSection:
1869 <ProductSection>
1870     <Property ovf:key="app.adminEmail" ovf:type="string" ovf:userConfigurable="true"
1871         ovf:configuration="standard">
1872         <Label>Admin email</Label>
1873         <Description>Email address of service administrator</Description>
1874     </Property>
1875     <Property ovf:key="app.log" ovf:type="string" ovf:value="low"
1876         ovf:userConfigurable="true">
1877         <Label>Loglevel</Label>
1878         <Description>Loglevel for the service</Description>
1879         <Value ovf:value="none" ovf:configuration="minimal">
1880     </Property>
1881 </ProductSection>

```

1882 In the example above, the app.adminEmail property is only user configurable in the
1883 standard configuration, while the default value for the app.log property is changed
1884 from low to none in the minimal configuration.

1885 D.17 Example of OperatingSystemSection

```

1886 <OperatingSystemSection ovf:id="76">
1887     <Info>Specifies the operating system installed</Info>
1888     <Description>Microsoft Windows Server 2008</Description>
1889 </OperatingSystemSection>

```

1890 D.18 Example of InstallSection

```

1891 <InstallSection ovf:initialBootStopDelay="300">

```

```

1892     <Info>Specifies that the virtual machine needs to be booted once after having
1893     created the guest software in order to install and/or configure the software
1894     </Info>
1895 </InstallSection>

```

1896 D.19 Example of EnvironmentFilesSection

1897 EXAMPLE:

```

1898 <Envelope>
1899     <References>
1900     ...
1901     <File ovf:id="config" ovf:href="config.xml" ovf:size="4332"/>
1902     <File ovf:id="resources" ovf:href="http://mywebsite/resources/resources.zip"/>
1903     </References>
1904     ...
1905     <VirtualSystem ovf:id="...">
1906     ...
1907     <ovf:EnvironmentFilesSection ovf:required="false" ovf:transport="iso">
1908         <Info>Config files to be included in OVF environment</Info>
1909         <ovf:File ovf:fileRef="config" ovf:path="setup/cfg.xml"/>
1910         <ovf:File ovf:fileRef="resources" ovf:path="setup/resources.zip"/>
1911     </ovf:EnvironmentFilesSection>
1912     ...
1913 </VirtualSystem>
1914 ...
1915 </Envelope>

```

1916 In the example above, the file config.xml in the OVF package will be copied to the OVF
 1917 environment ISO image and be accessible to the guest software in location
 1918 /ovffiles/setup/cfg.xml, while the file resources.zip will be accessible in location
 1919 /ovffiles/setup/resources.zip.

1920 D.20 Example of BootDeviceSection

1921 In the example below, the Pre-Install configuration specifies the boot source as a
 1922 specific device (network), while the Post-Install configuration specifies a device
 1923 type (hard disk).

1924 EXAMPLE:

```

1925 <Envelope>
1926 ...
1927 <VirtualSystem ovf:id="...">
1928 ...
1929 <ovf:BootDeviceSection>
1930     <Info>Boot device order specification</Info>
1931     <bootc:CIM_BootConfigSetting>
1932         <bootc:Caption>Pre-Install</bootc:Caption>
1933         <bootc:Description>Boot Sequence for fixup of disk</bootc:Description>
1934         <boots:CIM_BootSourceSetting>
1935             <boots:Caption>Fix-up DVD on the network</boots:Caption>
1936             <boots:InstanceID>3</boots:InstanceID>           <!-- Network device-->
1937         </boots:CIM_BootSourceSetting>
1938         <boots:CIM_BootSourceSetting>
1939             <boots:Caption>Boot virtual disk</boots:Caption>
1940             <boots:StructuredBootString>CIM:Hard-Disk</boots:StructuredBootString>
1941         </boots:CIM_BootSourceSetting>
1942     </bootc:CIM_BootConfigSetting>
1943 </ovf:BootDeviceSection>

```

```

1944 ...
1945     </VirtualSystem>
1946 </Envelope>

```

1947 D.21 Example of SharedDiskSection

1948 EXAMPLE:

```

1949 <ovf:SharedDiskSection>
1950     <Info>Describes the set of virtual disks shared between VMs</Info>
1951     <ovf:SharedDisk ovf:diskId="datadisk" ovf:fileRef="data"
1952         ovf:capacity="8589934592" ovf:populatedSize="3549324972"
1953         ovf:format=
1954             "http://www.vmware.com/interfaces/specifications/vmdk.html#sparse"/>
1955     <ovf:SharedDisk ovf:diskId="transientdisk" ovf:capacity="536870912"/>
1956 </ovf:SharedDiskSection>

```

1957 D.22 Example of ScaleOutSection

1958 EXAMPLE:

```

1959 <VirtualSystemCollection ovf:id="web-tier">
1960 ...
1961     <ovf:ScaleOutSection ovf:id="web-server">
1962         <Info>Web tier</Info>
1963         <ovf:Description>Number of web server instances in web tier</ovf:Description>
1964         <ovf:InstanceCount ovf:default="4" ovf:minimum="2" ovf:maximum="8"/>
1965     </ovf:ScaleOutSection>
1966 ...
1967     <VirtualSystem ovf:id="web-server">
1968         <Info>Prototype web server</Info>
1969     ...
1970 </VirtualSystem>
1971 </VirtualSystemCollection>

```

1972
1973 In the example above, the deployment platform creates a web tier containing between
1974 two and eight web server virtual machine instances, with a default instance count of
1975 four. The deployment platform makes an appropriate choice (e.g., by prompting the
1976 user). Assuming three replicas were created, the OVF environment available to the
1977 guest software in the first replica has the following content structure:

1978
1979 EXAMPLE:

```

1980 <Environment ... ovfenv:id="web-server-1">
1981     ...
1982     <Entity ovfenv:id="web-server-2">
1983         ...
1984     </Entity>
1985     <Entity ovfenv:id="web-server-3">
1986         ...
1987     </Entity>
1988 </Environment>

```

1989
1990 EXAMPLE:

```

1991 <VirtualSystemCollection ovf:id="web-tier">
1992     ...
1993     <DeploymentOptionSection>
1994         <Info>Deployment size options</Info>
1995         <Configuration ovf:id="minimal">

```

```

1996     <Label>Minimal</Label>
1997     <Description>Minimal deployment scenario</Description>
1998 </Configuration>
1999     <Configuration ovf:id="common" ovf:default="true">
2000     <Label>Typical</Label>
2001     <Description>Common deployment scenario</Description>
2002 </Configuration>
2003     ...
2004 </DeploymentOptionSection>
2005     ...
2006 <ovf:ScaleOutSection ovf:id="web-server">
2007     <Info>Web tier</Info>
2008     <ovf:Description>Number of web server instances in web tier</ovf:Description>
2009     <ovf:InstanceCount ovf:default="4"/>
2010     <ovf:InstanceCount ovf:default="1" ovf:configuration="minimal"/>
2011 </ovf:ScaleOutSection>
2012     ...
2013 </VirtualSystemCollection>

```

2014 In the example above, the default replica count is four, unless the minimal
 2015 deployment scenario is chosen, in which case the default is one.

2016 **D.23 Example of PlacementGroupSection**

2017 EXAMPLE:

```

2018 <Envelope ...>
2019     ...
2020     <ovf:PlacementGroupSection ovf:id="web" ovf:policy="availability">
2021     <Info>Placement policy for group of VMs</Info>
2022     <ovf:Description>Placement policy for web tier</ovf:Description>
2023 </ovf:PlacementGroupSection>
2024     ...
2025 <VirtualSystemCollection ovf:id="web-tier">
2026     ...
2027     <ovf:ScaleOutSection ovf:id="web-node">
2028     <Info>Web tier</Info>
2029     ...
2030 </ovf:ScaleOutSection>
2031     ...
2032 <VirtualSystem ovf:id="web-node">
2033     <Info>Web server</Info>
2034     ...
2035     <ovf:PlacementSection ovf:group="web">
2036     <Info>Placement policy group reference</Info>
2037 </ovf:PlacementSection>
2038     ...
2039 </VirtualSystem>
2040 </VirtualSystemCollection>
2041 </Envelope>

```

2042 In the example above, all virtual machines in the compute tier should be placed
 2043 separately for high availability. This example also use the ScaleOutSection defined in
 2044 clause 9.14, in which case each replica get the policy assigned.

2045 D.24 Example of EncryptionSection

2046 Below is an example of an OVF encryption section with encryption methods utilized in
 2047 the OVF document, and the corresponding reference list pointing to the items that have
 2048 been encrypted.

2049
 2050 EXAMPLE:

```

2051     <ovf:EncryptionSection>
2052 <!-- This section contains two different methods of encryption and the corresponding
2053 backpointers to the data that is encrypted ->
2054     <!-- Method#1: Pass phrase based Key derivation ->
2055 <!-- The following derived key block defines PBKDF2 and the corresponding back
2056 pointers to the encrypted data elements -->
2057     <!-- Use a salt value "ovfpassword" and iteration count of 4096 --->
2058     <xenc11:DerivedKey>
2059         <xenc11:KeyDerivationMethod
2060 Algorithm="http://www.rsasecurity.com/rsalabs/pkcs/schemas/pkcs-5#pbkdf2"/>
2061 <pkcs-5:PBKDF2-params>
2062         <Salt>
2063             <Specified>ovfpassword</Specified>
2064         </Salt>
2065         <IterationCount>4096</IterationCount>
2066         <KeyLength>16</KeyLength>
2067         <PRF Algorithm="http://www.w3.org/2001/04/xmldsig-more#hmac-
2068 sha256"/>
2069         </pkcs-5:PBKDF2-params>
2070     ...
2071 <!-- The ReferenceList element below contains references to the file Ref-109.vhd via
2072 the URI syntax which is specified by XML Encryption.
2073 --->
2074 <xenc:ReferenceList>
2075     <xenc:DataReference URI="#first.vhd" />
2076 <xenc:DataReference URI=... />
2077 <xenc:DataReference URI=... />
2078 </xenc:ReferenceList>
2079 </xenc11:DerivedKey>
2080     <!-- Method#2: The following example illustrates use of a symmetric key
2081 transported using the public key within a certificate ->
2082 <xenc:EncryptedKey>
2083     <xenc:EncryptionMethod
2084 Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
2085         <ds:KeyInfo xmlns:ds='http://www.w3.org/2000/09/xmldsig#'
2086             <ds:X509Data>
2087                 <ds:X509Certificate> ... </ds:X509Certificate>
2088             </ds:X509Data>
2089             </ds:KeyInfo>
2090     <xenc:CipherData>
2091     <xenc:CipherValue> ... </xenc:CipherValue>
2092     </xenc:CipherData>
2093 <!-- The ReferenceList element below contains reference #second-xml-fragment" to the
2094 XML fragment that has been encrypted using the above method --->
2095 <xenc:ReferenceList>
2096     <xenc:DataReference URI='#second-xml-fragment' />
2097     <xenc:DataReference URI='...' />
2098     <xenc:DataReference URI='...' />
2099 </xenc:ReferenceList>

```

```

2100     </xenc:EncryptedKey>
2101 </ovf:EncryptionSection>
2102 Below is an example of the encrypted file which is referenced in the EncryptionSection
2103 above using URI='Ref-109.vhd' syntax.
2104 EXAMPLE:
2105 <ovf:References>
2106 <ovf:File ovf:id="Xen:9cb10691-4012-4aeb-970c-3d47a906bfff/0b13bdba-3761-8622-22fc-
2107 2e252ed9ce14" ovf:href="Ref-109.vhd">
2108 <!-- the encrypted file referenced by the package is enclosed by an EncryptedData with
2109 a CipherReference to the actual encrypted file. The EncryptionSection in this example
2110 has a back pointer to it under the PBKDF2 algorithm via Id="first.vhd". This tells the
2111 decrypter how to decrypt the file -->
2112 <xenc:EncryptedData Id="first.vhd" Type='http://www.w3.org/2001/04/xmlenc#Element' >
2113     <xenc:EncryptionMethod
2114 Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc" />
2115     <xenc:CipherData>
2116         <xenc:CipherReference URI='Ref-109.vhd' />
2117     </xenc:CipherData>
2118 </xenc:EncryptedData>
2119 </ovf:File>
2120 </ovf:References>
2121 Below is an example of the encrypted OVF markup which is referenced in the
2122 EncryptionSection above using URI='#second-xml-fragment' syntax.
2123 EXAMPLE:
2124 <!-- the EncryptedData element below encompasses encrypted xml from the original
2125 document. It is provided with the Id "first-xml-fragment" which allows it to be
2126 referenced from the EncryptionSection. -->
2127 <xenc:EncryptedData Type=http://www.w3.org/2001/04/xmlenc#Element Id="second-xml-
2128 fragment">
2129 <!-- Each EncryptedData specifies its own encryption method. -->
2130     <xenc:EncryptionMethod Algorithm=http://www.w3.org/2001/04-xmlenc#aes128-cbc/>
2131     <xenc:CipherData>
2132         <!-- Encrypted content --->
2133         <xenc:CipherValue>DEADBEEF</xenc:CipherValue>
2134     </xenc:CipherData>
2135 </xenc:EncryptedData>

```

2136 D.25 Example of Internationalization

```

2137 EXAMPLE 1:
2138 <Info ovf:msgid="info.text">Default info.text value if no locale is set or no locale
2139 match</Info>
2140 <License ovf:msgid="license.tomcat-6_0"/> <!-- No default message -->
2141

```

2142 Using Internal Resource Bundles

```

2143
2144 EXAMPLE 2:
2145 <ovf:Envelope xml:lang="en-US">
2146     ...
2147     ... sections and content here ...
2148     ...
2149     <Info msgid="info.os">Operating System</Info>
2150     ...
2151     <Strings xml:lang="da-DA">
2152         <Msg ovf:msgid="info.os">Operativsystem</Msg>
2153     ...

```

```

2154     </Strings>
2155     <Strings xml:lang="de-DE">
2156         <Msg ovf:msgid="info.os">Betriebssystem</Msg>
2157         ...
2158     </Strings>
2159 </ovf:Envelope>
2160

```

10.2 External Resource Bundles

EXAMPLE 3:

```

2163 <ovf:Envelope xml:lang="en-US">
2164     <References>
2165         ...
2166         <File ovf:id="it-it-resources" ovf:href="resources/it-it-bundle.msg"/>
2167     </References>
2168     ... sections and content here ...
2169     ...
2170     <Strings xml:lang="it-IT" ovf:fileRef="it-it-resources"/>
2171     ...
2172 </ovf:Envelope>

```

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

```

2175 <Strings
2176     xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1"
2177     xmlns="http://schemas.dmtf.org/ovf/envelope/1"
2178     xml:lang="it-IT">
2179     <Msg ovf:msgid="info.os">Sistema operativo</Msg>
2180     ...
2181 </Strings>

```

D.26 Example of Message Content in an External File

EXAMPLE:

```

2184 <Envelope xml:lang="en-US">
2185     <References>
2186         <File ovf:id="license-en-US" ovf:href="license-en-US.txt"/>
2187         <File ovf:id="license-de-DE" ovf:href="license-de-DE.txt"/>
2188     </References>
2189     ...
2190 <VirtualSystem ovf:id="...">
2191     <EulaSection>
2192         <Info>Licensing agreement</Info>
2193         <License ovf:msgid="license">Unused</License>
2194     </EulaSection>
2195     ...
2196 </VirtualSystem>
2197     ...
2198 <Strings xml:lang="en-US">
2199     <Msg ovf:msgid="license" ovf:fileRef="license-en-US">Invalid license</Msg>
2200 </Strings>
2201 <Strings xml:lang="de-DE">
2202     <Msg ovf:msgid="license" ovf:fileRef="license-de-DE">Ihre Lizenz ist nicht
2203 gültig</Msg>
2204 </Strings>
2205 </Envelope>

```

2206 In the example above, the default license agreement is stored in plain text file
 2207 license-en-US.txt, while the license agreement for the de-DE locale is stored in file
 2208 license-de-DE.txt.
 2209 Note that the above mechanism works for all localizable elements and not just License.

2210 **D.27 Example of Environment Document**

2211 EXAMPLE: An example of the structure of the OVF environment document follows:
 2212 `<?xml version="1.0" encoding="UTF-8"?>`
 2213 `<Environment xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"`
 2214 `xmlns:ovfenv="http://schemas.dmtf.org/ovf/environment/1"`
 2215 `xmlns="http://schemas.dmtf.org/ovf/environment/1"`
 2216 `ovfenv:id="identification of VM from OVF descriptor">`
 2217 `<!-- Information about virtualization platform -->`
 2218 `<PlatformSection>`
 2219 `<Kind>Type of virtualization platform</Kind>`
 2220 `<Version>Version of virtualization platform</Version>`
 2221 `<Vendor>Vendor of virtualization platform</Vendor>`
 2222 `<Locale>Language and country code</Locale>`
 2223 `<TimeZone>Current timezone offset in minutes from UTC</TimeZone>`
 2224 `</PlatformSection>`
 2225 `<!-- Properties defined for this virtual machine -->`
 2226 `<PropertySection>`
 2227 `<Property ovfenv:key="key" ovfenv:value="value">`
 2228 `<!-- More properties -->`
 2229 `</PropertySection>`
 2230 `<Entity ovfenv:id="id of sibling virtual system or virtual system collection">`
 2231 `<PropertySection>`
 2232 `<!-- Properties from sibling -->`
 2233 `</PropertySection>`
 2234 `</Entity>`
 2235 `</Environment>`

ANNEX E (informative)

Network Port Profile Examples

E.1 Example 1 (OVF Descriptor for One Virtual System and One Network with an Inlined Network Port Profile)

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

```

2247 <?xml version="1.0" encoding="UTF-8"?>
2248 <Envelope xsi:schemaLocation="http://schemas.dmtf.org/ovf/envelope/2
2249 file:///C:/dsp8023_2.0.0_wgv0.9.5.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2250 xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/2" xmlns="http://schemas.dmtf.org/ovf/envelope/2"
2251 xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
2252 xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
2253 xmlns:epasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2254 schema/2/CIM_EthernetPortAllocationSettingData"
2255 xmlns:sasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_StorageAllocationSettingData">
2256 <!-- References to all external files -->
2257 <References>
2258 <File ovf:id="file1" ovf:href="vmdisk1.vmdk" ovf:size="2000000000"/>
2259 </References>
2260 <!-- Describes meta-information for all virtual disks in the package -->
2261 <DiskSection>
2262 <Info>Describes the set of virtual disks</Info>
2263 <Disk ovf:diskId="vmdisk1" ovf:fileRef="file1" ovf:capacity="4294967296"
2264 ovf:format="http://www.examplecompany.com/interfaces/specifications/vmdk.html#sparse"/>
2265 </DiskSection>
2266 <!-- Describes all networks used in the package -->
2267 <NetworkSection>
2268 <Info>List of logical networks used in the package</Info>
2269 <Network ovf:name="VM Network">
2270 <Description>The network that the VMs connect to</Description>
2271 <NetworkPortProfile>
2272 <!-- Network port profile describing bandwidth reservation. Network port profile
2273 is identified by UUID. -->
2274 <Item>
2275 <epasd:AllocationUnits>bit / second * 10^9</epasd:AllocationUnits>
2276 <epasd:ElementName>Network Port Profile 1</epasd:ElementName>
2277 <epasd:InstanceID>1</epasd:InstanceID>
2278 <epasd:NetworkPortProfileID>aaaaaaaa-bbbb-cccc-dddd-
2279 eeeeeeeeeee</epasd:NetworkPortProfileID>
2280 <epasd:NetworkPortProfileIDType>3</epasd:NetworkPortProfileIDType>
2281 <epasd:Reservation>1</epasd:Reservation>
2282 </Item>
2283 </NetworkPortProfile>
2284 </Network>
2285 </NetworkSection>
2286 <VirtualSystem ovf:id="vm">
2287 <Info>Describes a virtual machine</Info>
2288 <Name>Virtual Appliance One</Name>
2289 <ProductSection>
2290 <Info>Describes product information for the appliance</Info>
2291 <Product>The Great Appliance</Product>
2292 <Vendor>Some Great Corporation</Vendor>
2293 <Version>13.00</Version>
2294 <FullVersion>13.00-b5</FullVersion>
2295 <ProductUrl>http://www.somegreatcorporation.com/greatappliance</ProductUrl>
2296 <VendorUrl>http://www.somegreatcorporation.com/</VendorUrl>
2297 <Property ovf:key="admin.email" ovf:type="string">

```

```

2298         <Description>Email address of administrator</Description>
2299     </Property>
2300     <Property ovf:key="app.ip" ovf:type="string" ovf:defaultValue="192.168.0.10">
2301         <Description>The IP address of this appliance</Description>
2302     </Property>
2303 </ProductSection>
2304 <AnnotationSection ovf:required="false">
2305     <Info>A random annotation on this service. It can be ignored</Info>
2306     <Annotation>Contact customer support if you have any problems</Annotation>
2307 </AnnotationSection>
2308 <EulaSection>
2309     <Info>License information for the appliance</Info>
2310     <License>Insert your favorite license here</License>
2311 </EulaSection>
2312 <VirtualHardwareSection>
2313     <Info>Memory = 4 GB, CPU = 1 GHz, Disk = 100 GB, 1 Ethernet nic</Info>
2314     <Item>
2315         <rasd:AllocationUnits>Hertz*10^9</rasd:AllocationUnits>
2316         <rasd:Description>Virtual CPU</rasd:Description>
2317         <rasd:ElementName>1 GHz virtual CPU</rasd:ElementName>
2318         <rasd:InstanceID>1</rasd:InstanceID>
2319         <rasd:Reservation>1</rasd:Reservation>
2320         <rasd:ResourceType>3</rasd:ResourceType>
2321         <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2322     </Item>
2323     <Item>
2324         <rasd:AllocationUnits>byte*2^30</rasd:AllocationUnits>
2325         <rasd:Description>Memory</rasd:Description>
2326         <rasd:ElementName>1 GByte of memory</rasd:ElementName>
2327         <rasd:InstanceID>2</rasd:InstanceID>
2328         <rasd:ResourceType>4</rasd:ResourceType>
2329         <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2330     </Item>
2331     <EthernetPortItem>
2332         <epasd:Address>00-16-8B-DB-00-5E</epasd:Address>
2333         <epasd:AllocationUnits>bit / second * 10^9 </epasd:AllocationUnits>
2334         <epasd:Connection>VM Network</epasd:Connection>
2335         <epasd:Description>Virtual NIC</epasd:Description>
2336         <epasd:ElementName>Ethernet Port</epasd:ElementName>
2337
2338         <epasd:InstanceID>3</epasd:InstanceID>
2339         <epasd:NetworkPortProfileID>aaaaaaa-bbbb-cccc-dddd-
2340 eeeeeeeee</epasd:NetworkPortProfileID>
2341         <epasd:NetworkPortProfileIDType>3</epasd:NetworkPortProfileIDType>
2342         <epasd:Reservation>1</epasd:Reservation>
2343         <epasd:ResourceType>10</epasd:ResourceType>
2344         <epasd:VirtualQuantityUnits>1</epasd:VirtualQuantityUnits>
2345     </EthernetPortItem>
2346     <StorageItem>
2347         <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
2348         <sasd:Description>Virtual Disk</sasd:Description>
2349         <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
2350         <sasd:InstanceID>4</sasd:InstanceID>
2351         <sasd:Reservation>100</sasd:Reservation>
2352         <sasd:ResourceType>31</sasd:ResourceType>
2353         <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
2354     </StorageItem>
2355 </VirtualHardwareSection>
2356 <OperatingSystemSection ovf:id="58" ovf:required="false">
2357     <Info>Guest Operating System</Info>
2358     <Description>OS</Description>
2359 </OperatingSystemSection>
2360 </VirtualSystem>
2361 </Envelope>

```

2362 E.2 Example 2 (OVF Descriptor for One Virtual System and One Network with a 2363 Locally Referenced Network Port Profile)

2364 The example below shows an OVF descriptor that describes a virtual system and a network it connects
2365 to. The virtual system description in this example uses a network port profile that is described in a local
2366 file that is contained in the same OVF package. The network described in the network section uses the
2367 same network port profile description. The network port profile described in this example is used to
2368 reserve 1 Gbps of bandwidth.

```

2369 <?xml version="1.0" encoding="UTF-8"?>
2370 <Envelope xsi:schemaLocation="http://schemas.dmtf.org/ovf/envelope/2
2371 file:///C:/dsp8023_2.0.0_wgv0.9.5.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2372 xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/2" xmlns="http://schemas.dmtf.org/ovf/envelope/2"
2373 xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
2374 xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
2375 xmlns:epasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2376 schema/2/CIM_EthernetPortAllocationSettingData"
2377 xmlns:sasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_StorageAllocationSettingData">
2378 <!-- References to all external files -->
2379 <References>
2380 <File ovf:id="file1" ovf:href="vmdisk1.vmdk" ovf:size="2000000000"/>
2381 <File ovf:id="networkportprofile1" ovf:href="NetworkPortProfile1.xml"/>
2382 </References>
2383 <!-- Describes meta-information for all virtual disks in the package -->
2384 <DiskSection>
2385 <Info>Describes the set of virtual disks</Info>
2386 <Disk ovf:diskId="vmdisk1" ovf:fileRef="file1" ovf:capacity="4294967296"
2387 ovf:format="http://www.examplecompany.com/interfaces/specifications/vmdk.html#sparse"/>
2388 </DiskSection>
2389 <!-- Describes all networks used in the package -->
2390 <NetworkSection>
2391 <Info>List of logical networks used in the package</Info>
2392 <Network ovf:name="VM Network">
2393 <Description>The network that VMs connect to</Description>
2394 <NetworkPortProfileURI>file:networkportprofile1</NetworkPortProfileURI>
2395 </Network>
2396 </NetworkSection>
2397 <VirtualSystem ovf:id="vm">
2398 <Info>Describes a virtual machine</Info>
2399 <Name>Virtual Appliance One</Name>
2400 <ProductSection>
2401 <Info>Describes product information for the appliance</Info>
2402 <Product>The Great Appliance</Product>
2403 <Vendor>Some Great Corporation</Vendor>
2404 <Version>13.00</Version>
2405 <FullVersion>13.00-b5</FullVersion>
2406 <ProductUrl>http://www.somegreatcorporation.com/greatappliance</ProductUrl>
2407 <VendorUrl>http://www.somegreatcorporation.com/</VendorUrl>
2408 <Property ovf:key="admin.email" ovf:type="string">
2409 <Description>Email address of administrator</Description>
2410 </Property>
2411 <Property ovf:key="app.ip" ovf:type="string" ovf:defaultValue="192.168.0.10">
2412 <Description>The IP address of this appliance</Description>
2413 </Property>
2414 </ProductSection>
2415 <AnnotationSection ovf:required="false">
2416 <Info>A random annotation on this service. It can be ignored</Info>
2417 <Annotation>Contact customer support if you have any problems</Annotation>
2418 </AnnotationSection>
2419 <EulaSection>
2420 <Info>License information for the appliance</Info>
2421 <License>Insert your favorite license here</License>
2422 </EulaSection>
2423 <VirtualHardwareSection>
2424 <Info>Memory = 4 GB, CPU = 1 GHz, Disk = 100 GB, 1 Ethernet nic</Info>
2425 <Item>
2426 <rasd:AllocationUnits>Hertz*10^9</rasd:AllocationUnits>
2427 <rasd:Description>Virtual CPU</rasd:Description>
2428 <rasd:ElementName>1 GHz virtual CPU</rasd:ElementName>

```

```

2429         <rasd:InstanceID>1</rasd:InstanceID>
2430         <rasd:Reservation>1</rasd:Reservation>
2431         <rasd:ResourceType>3</rasd:ResourceType>
2432         <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2433     </Item>
2434 </Item>
2435     <rasd:AllocationUnits>byte*2^30</rasd:AllocationUnits>
2436     <rasd:Description>Memory</rasd:Description>
2437     <rasd:ElementName>1 GByte of memory</rasd:ElementName>
2438     <rasd:InstanceID>2</rasd:InstanceID>
2439     <rasd:ResourceType>4</rasd:ResourceType>
2440     <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2441 </Item>
2442 <EthernetPortItem>
2443     <epasd:Address>00-16-8B-DB-00-5E</epasd:Address>
2444     <epasd:Connection>VM Network</epasd:Connection>
2445     <epasd:Description>Virtual NIC</epasd:Description>
2446     <epasd:ElementName>Ethernet Port</epasd:ElementName>
2447
2448     <epasd:InstanceID>3</epasd:InstanceID>
2449     <epasd:NetworkPortProfileID>file:networkportprofile1</epasd:NetworkPortProfileID>
2450     <epasd:NetworkPortProfileIDType>2</epasd:NetworkPortProfileIDType>
2451     <epasd:ResourceType>10</epasd:ResourceType>
2452     <epasd:VirtualQuantityUnits>1</epasd:VirtualQuantityUnits>
2453 </EthernetPortItem>
2454 <StorageItem>
2455     <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
2456     <sasd:Description>Virtual Disk</sasd:Description>
2457     <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
2458     <sasd:InstanceID>4</sasd:InstanceID>
2459     <sasd:Reservation>100</sasd:Reservation>
2460     <sasd:ResourceType>31</sasd:ResourceType>
2461     <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
2462 </StorageItem>
2463 </VirtualHardwareSection>
2464 <OperatingSystemSection ovf:id="58" ovf:required="false">
2465     <Info>Guest Operating System</Info>
2466     <Description>OS</Description>
2467 </OperatingSystemSection>
2468 </VirtualSystem>
2469 </Envelope>

```

2470 **E.3 Example 3 (OVF Descriptor for One Virtual System and One Network with a** 2471 **Network Port Profile referenced by a URI)**

2472 The example below shows an OVF descriptor that describes a virtual system and a network it connects
2473 to. The virtual system description in this example uses a network port profile that is described by a URI.
2474 The network described in the network section uses the same network port profile description. The
2475 network port profile described in this example is used to reserve 1 Gbps of bandwidth.

```

2476 <?xml version="1.0" encoding="UTF-8"?>
2477 <Envelope xsi:schemaLocation="http://schemas.dmtf.org/ovf/envelope/2
2478 file:///C:/dsp8023 2.0.0 wgv0.9.5.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2479 xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/2" xmlns="http://schemas.dmtf.org/ovf/envelope/2"
2480 xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
2481 xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
2482 xmlns:epasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2483 schema/2/CIM_EthernetPortAllocationSettingData"
2484 xmlns:sasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_StorageAllocationSettingData">
2485 <!-- References to all external files -->
2486 <References>
2487     <File ovf:id="file1" ovf:href="vmdisk1.vmdk" ovf:size="2000000000"/>
2488 </References>
2489 <!-- Describes meta-information for all virtual disks in the package -->
2490 <DiskSection>
2491     <Info>Describes the set of virtual disks</Info>
2492     <Disk ovf:diskId="vmdisk1" ovf:fileRef="file1" ovf:capacity="4294967296"
2493 ovf:format="http://www.examplecompany.com/interfaces/specifications/vmdk.html#sparse"/>
2494 </DiskSection>

```

```

2495 <!-- Describes all networks used in the package -->
2496 <NetworkSection>
2497   <Info>List of logical networks used in the package</Info>
2498   <Network ovf:name="VM Network">
2499     <Description>The network that the VMs connect to</Description>
2500
2501   <NetworkPortProfileURI>http://www.dmtf.org/networkportprofiles/networkportprofile1.xml</Netwo
2502 rkPortProfileURI>
2503   </Network>
2504 </NetworkSection>
2505 <VirtualSystem ovf:id="vm">
2506   <Info>Describes a virtual machine</Info>
2507   <Name>Virtual Appliance One</Name>
2508   <ProductSection>
2509     <Info>Describes product information for the appliance</Info>
2510     <Product>The Great Appliance</Product>
2511     <Vendor>Some Great Corporation</Vendor>
2512     <Version>13.00</Version>
2513     <FullVersion>13.00-b5</FullVersion>
2514     <ProductUrl>http://www.somegreatcorporation.com/greatappliance</ProductUrl>
2515     <VendorUrl>http://www.somegreatcorporation.com/</VendorUrl>
2516     <Property ovf:key="admin.email" ovf:type="string">
2517       <Description>Email address of administrator</Description>
2518     </Property>
2519     <Property ovf:key="app.ip" ovf:type="string" ovf:defaultValue="192.168.0.10">
2520       <Description>The IP address of this appliance</Description>
2521     </Property>
2522   </ProductSection>
2523   <AnnotationSection ovf:required="false">
2524     <Info>A random annotation on this service. It can be ignored</Info>
2525     <Annotation>Contact customer support if you have any problems</Annotation>
2526   </AnnotationSection>
2527   <EulaSection>
2528     <Info>License information for the appliance</Info>
2529     <License>Insert your favorite license here</License>
2530   </EulaSection>
2531   <VirtualHardwareSection>
2532     <Info>Memory = 4 GB, CPU = 1 GHz, Disk = 100 GB, 1 Ethernet nic</Info>
2533     <Item>
2534       <rasd:AllocationUnits>Hertz*10^9</rasd:AllocationUnits>
2535       <rasd:Description>Virtual CPU</rasd:Description>
2536       <rasd:ElementName>1 GHz virtual CPU</rasd:ElementName>
2537       <rasd:InstanceID>1</rasd:InstanceID>
2538       <rasd:Reservation>1</rasd:Reservation>
2539       <rasd:ResourceType>3</rasd:ResourceType>
2540       <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2541     </Item>
2542     <Item>
2543       <rasd:AllocationUnits>byte*2^30</rasd:AllocationUnits>
2544       <rasd:Description>Memory</rasd:Description>
2545       <rasd:ElementName>1 GByte of memory</rasd:ElementName>
2546       <rasd:InstanceID>2</rasd:InstanceID>
2547       <rasd:ResourceType>4</rasd:ResourceType>
2548       <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2549     </Item>
2550     <EthernetPortItem>
2551       <epasd:Address>00-16-8B-DB-00-5E</epasd:Address>
2552       <epasd:Connection>VM Network</epasd:Connection>
2553       <epasd:Description>Virtual NIC</epasd:Description>
2554       <epasd:ElementName>Ethernet Port</epasd:ElementName>
2555
2556       <epasd:InstanceID>3</epasd:InstanceID>
2557
2558       <epasd:NetworkPortProfileID>http://www.dmtf.org/networkportprofiles/networkportprofile1.xml</
2559 epasd:NetworkPortProfileID>
2560       <epasd:NetworkPortProfileIDType>2</epasd:NetworkPortProfileIDType>
2561       <epasd:ResourceType>10</epasd:ResourceType>
2562       <epasd:VirtualQuantityUnits>1</epasd:VirtualQuantityUnits>
2563     </EthernetPortItem>
2564     <StorageItem>
2565       <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>

```

```

2566     <sas:Description>Virtual Disk</sas:Description>
2567     <sas:ElementName>100 GByte Virtual Disk</sas:ElementName>
2568     <sas:InstanceID>4</sas:InstanceID>
2569     <sas:Reservation>100</sas:Reservation>
2570     <sas:ResourceType>31</sas:ResourceType>
2571     <sas:VirtualQuantity>1</sas:VirtualQuantity>
2572   </StorageItem>
2573 </VirtualHardwareSection>
2574 <OperatingSystemSection ovf:id="58" ovf:required="false">
2575   <Info>Guest Operating System</Info>
2576   <Description>OS</Description>
2577 </OperatingSystemSection>
2578 </VirtualSystem>
2579 </Envelope>

```

2580 E.4 Example 4 (OVF Descriptor for Two Virtual Systems and One Network with 2581 Two Network Port Profiles referenced by URIs)

2582 The example below shows an OVF descriptor that describes two virtual systems and a network they
2583 connect to. Each virtual system description in this example uses a network port profile that is described
2584 by a URI. The network described in the network section uses the same two network port profiles. The two
2585 network port profiles described in this example are used to reserve 1 Gbps of bandwidth and describe
2586 general network traffic respectively. Annex E.5 and E.6 are examples of these network port profiles.

```

2587 <?xml version="1.0" encoding="UTF-8"?>
2588 <Envelope xsi:schemaLocation="http://schemas.dmtf.org/ovf/envelope/2
2589 file:///C:/dsp8023_2.0.0_wgv0.9.5.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2590 xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/2" xmlns="http://schemas.dmtf.org/ovf/envelope/2"
2591 xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
2592 xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
2593 xmlns:epasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2594 schema/2/CIM_EthernetPortAllocationSettingData"
2595 xmlns:sasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_StorageAllocationSettingData">
2596 <!-- References to all external files -->
2597 <References>
2598   <File ovf:id="file1" ovf:href="vmdisk1.vmdk" ovf:size="2000000000"/>
2599 </References>
2600 <!-- Describes meta-information for all virtual disks in the package -->
2601 <DiskSection>
2602   <Info>Describes the set of virtual disks</Info>
2603   <Disk ovf:diskId="vmdisk1" ovf:fileRef="file1" ovf:capacity="4294967296"
2604 ovf:format="http://www.examplecompany.com/interfaces/specifications/vmdk.html#sparse"/>
2605 </DiskSection>
2606 <!-- Describes all networks used in the package -->
2607 <NetworkSection>
2608   <Info>List of logical networks used in the package</Info>
2609   <Network ovf:name="VM Network">
2610     <Description>The network that the VMs connect to</Description>
2611     <!-- Network port profile for storage traffic -->
2612
2613     <NetworkPortProfileURI>http://www.dmtf.org/networkportprofiles/networkportprofile1.xml</Netwo
2614 rkPortProfileURI>
2615     <!-- Network port profile for networking traffic -->
2616
2617     <NetworkPortProfileURI>http://www.dmtf.org/networkportprofiles/networkportprofile2.xml</Netwo
2618 rkPortProfileURI>
2619   </Network>
2620 </NetworkSection>
2621 <VirtualSystemCollection ovf:id="vscl">
2622   <Info>Collection of 2 VMs</Info>
2623   <VirtualSystem ovf:id="storage server">
2624     <Info>Describes a virtual machine</Info>
2625     <Name>Virtual Appliance One</Name>
2626     <ProductSection>
2627       <Info>Describes product information for the appliance</Info>
2628       <Product>The Great Appliance</Product>
2629       <Vendor>Some Great Corporation</Vendor>
2630       <Version>13.00</Version>
2631       <FullVersion>13.00-b5</FullVersion>

```

```

2632     <ProductUrl>http://www.somegreatcorporation.com/greatappliance</ProductUrl>
2633     <VendorUrl>http://www.somegreatcorporation.com/</VendorUrl>
2634     <Property ovf:key="admin.email" ovf:type="string">
2635         <Description>Email address of administrator</Description>
2636     </Property>
2637     <Property ovf:key="app.ip" ovf:type="string" ovf:defaultValue="192.168.0.10">
2638         <Description>The IP address of this appliance</Description>
2639     </Property>
2640 </ProductSection>
2641 <AnnotationSection ovf:required="false">
2642     <Info>A random annotation on this service. It can be ignored</Info>
2643     <Annotation>Contact customer support if you have any problems</Annotation>
2644 </AnnotationSection>
2645 <EulaSection>
2646     <Info>License information for the appliance</Info>
2647     <License>Insert your favorite license here</License>
2648 </EulaSection>
2649 <VirtualHardwareSection>
2650     <Info>Memory = 4 GB, CPU = 1 GHz, Disk = 100 GB, 1 Ethernet nic</Info>
2651     <Item>
2652         <rasd:AllocationUnits>Hertz*10^9</rasd:AllocationUnits>
2653         <rasd:Description>Virtual CPU</rasd:Description>
2654         <rasd:ElementName>1 GHz virtual CPU</rasd:ElementName>
2655         <rasd:InstanceID>1</rasd:InstanceID>
2656         <rasd:Reservation>1</rasd:Reservation>
2657         <rasd:ResourceType>3</rasd:ResourceType>
2658         <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2659     </Item>
2660     <Item>
2661         <rasd:AllocationUnits>byte*2^30</rasd:AllocationUnits>
2662         <rasd:Description>Memory</rasd:Description>
2663         <rasd:ElementName>1 GByte of memory</rasd:ElementName>
2664         <rasd:InstanceID>2</rasd:InstanceID>
2665         <rasd:ResourceType>4</rasd:ResourceType>
2666         <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2667     </Item>
2668     <EthernetPortItem>
2669         <epasd:Address>00-16-8B-DB-00-5E</epasd:Address>
2670         <epasd:Connection>VM Network</epasd:Connection>
2671         <epasd:Description>Virtual NIC</epasd:Description>
2672
2673         <epasd:ElementName>Ethernet Port</epasd:ElementName>
2674
2675         <epasd:InstanceID>3</epasd:InstanceID>
2676
2677         <epasd:NetworkPortProfileID>http://www.dmtf.org/networkportprofiles/networkportprofile1.xml</
2678     epasd:NetworkPortProfileID>
2679         <epasd:NetworkPortProfileIDType>2</epasd:NetworkPortProfileIDType>
2680         <epasd:ResourceType>10</epasd:ResourceType>
2681         <epasd:VirtualQuantityUnits>1</epasd:VirtualQuantityUnits>
2682     </EthernetPortItem>
2683     <StorageItem>
2684         <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
2685         <sasd:Description>Virtual Disk</sasd:Description>
2686         <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
2687         <sasd:InstanceID>4</sasd:InstanceID>
2688         <sasd:Reservation>100</sasd:Reservation>
2689         <sasd:ResourceType>31</sasd:ResourceType>
2690         <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
2691     </StorageItem>
2692 </VirtualHardwareSection>
2693 <OperatingSystemSection ovf:id="58" ovf:required="false">
2694     <Info>Guest Operating System</Info>
2695     <Description>OS</Description>
2696 </OperatingSystemSection>
2697 </VirtualSystem>
2698 <VirtualSystem ovf:id="web-server">
2699     <Info>Describes a virtual machine</Info>
2700     <Name>Virtual Appliance Two</Name>
2701     <ProductSection>
2702     <Info>Describes product information for the appliance</Info>

```

```

2703     <Product>The Great Appliance</Product>
2704     <Vendor>Some Great Corporation</Vendor>
2705     <Version>13.00</Version>
2706     <FullVersion>13.00-b5</FullVersion>
2707     <ProductUrl>http://www.somegreatcorporation.com/greatappliance</ProductUrl>
2708     <VendorUrl>http://www.somegreatcorporation.com/</VendorUrl>
2709     <Property ovf:key="admin.email" ovf:type="string">
2710         <Description>Email address of administrator</Description>
2711     </Property>
2712     <Property ovf:key="app.ip" ovf:type="string" ovf:defaultValue="192.168.0.10">
2713         <Description>The IP address of this appliance</Description>
2714     </Property>
2715 </ProductSection>
2716 <AnnotationSection ovf:required="false">
2717     <Info>A random annotation on this service. It can be ignored</Info>
2718     <Annotation>Contact customer support if you have any problems</Annotation>
2719 </AnnotationSection>
2720 <EulaSection>
2721     <Info>License information for the appliance</Info>
2722     <License>Insert your favorite license here</License>
2723 </EulaSection>
2724 <VirtualHardwareSection>
2725     <Info>Memory = 4 GB, CPU = 1 GHz, Disk = 100 GB, 1 Ethernet nic</Info>
2726     <Item>
2727         <rasd:AllocationUnits>Hertz*10^9</rasd:AllocationUnits>
2728         <rasd:Description>Virtual CPU</rasd:Description>
2729         <rasd:ElementName>1 GHz virtual CPU</rasd:ElementName>
2730         <rasd:InstanceID>1</rasd:InstanceID>
2731         <rasd:Reservation>1</rasd:Reservation>
2732         <rasd:ResourceType>3</rasd:ResourceType>
2733         <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2734     </Item>
2735     <Item>
2736         <rasd:AllocationUnits>byte*2^30</rasd:AllocationUnits>
2737         <rasd:Description>Memory</rasd:Description>
2738         <rasd:ElementName>1 GByte of memory</rasd:ElementName>
2739         <rasd:InstanceID>2</rasd:InstanceID>
2740         <rasd:ResourceType>4</rasd:ResourceType>
2741         <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2742     </Item>
2743     <EthernetPortItem>
2744         <epasd:Address>00-16-8B-DB-00-5F</epasd:Address>
2745         <epasd:Connection>VM Network</epasd:Connection>
2746         <epasd:Description>Virtual NIC</epasd:Description>
2747
2748         <epasd:ElementName>Ethernet Port</epasd:ElementName>
2749         <!-- Virtual NIC for networking traffic -->
2750         <epasd:InstanceID>3</epasd:InstanceID>
2751
2752         <epasd:NetworkPortProfileID>http://www.dmtf.org/networkportprofiles/networkportprofile2.xml</
2753 epasd:NetworkPortProfileID>
2754         <epasd:NetworkPortProfileIDType>2</epasd:NetworkPortProfileIDType>
2755         <epasd:ResourceType>10</epasd:ResourceType>
2756         <epasd:VirtualQuantityUnits>1</epasd:VirtualQuantityUnits>
2757     </EthernetPortItem>
2758     <StorageItem>
2759         <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
2760         <sasd:Description>Virtual Disk</sasd:Description>
2761         <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
2762         <sasd:InstanceID>4</sasd:InstanceID>
2763         <sasd:Reservation>100</sasd:Reservation>
2764         <sasd:ResourceType>31</sasd:ResourceType>
2765         <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
2766     </StorageItem>
2767 </VirtualHardwareSection>
2768 <OperatingSystemSection ovf:id="58" ovf:required="false">
2769     <Info>Guest Operating System</Info>
2770     <Description>OS</Description>
2771 </OperatingSystemSection>
2772 </VirtualSystem>
2773 </VirtualSystemCollection>

```

2774 </Envelope>

2775 E.5 Example 5 (networkportprofile1.xml)

2776

2777 Network Port profile example for bandwidth reservation.

```

2778 <?xml version="1.0" encoding="UTF-8"?>
2779 <NetworkPortProfile xsi:schemaLocation="http://schemas.dmtf.org/ovf/networkportprofile/1
2780 http://schemas.dmtf.org/ovf/networkportprofile/1/dsp8049.xsd"
2781 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2782 xmlns="http://schemas.dmtf.org/ovf/networkportprofile/1"
2783 xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
2784 xmlns:epasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2785 schema/2/CIM_EthernetPortAllocationSettingData">
2786   <Item>
2787     <epasd:AllocationUnits>bit / second * 10^9</epasd:AllocationUnits>
2788     <epasd:ElementName>Network Port Profile 1</epasd:ElementName>
2789     <epasd:InstanceID>1</epasd:InstanceID>
2790     <epasd:NetworkPortProfileID>aaaaaaaa-bbbb-cccc-dddd-
2791 eeeeeeeeeeee</epasd:NetworkPortProfileID>
2792     <epasd:NetworkPortProfileIDType>3</epasd:NetworkPortProfileIDType>
2793     <epasd:Reservation>1</epasd:Reservation>
2794   </Item>
2795 </NetworkPortProfile>

```

2796 E.6 Example 6 (networkportprofile2.xml)

2797

2798 Network Port Profile example showing priority setting.

```

2799 <?xml version="1.0" encoding="UTF-8"?>
2800 <NetworkPortProfile xsi:schemaLocation="http://schemas.dmtf.org/ovf/networkportprofile/1
2801 http://schemas.dmtf.org/ovf/networkportprofile/1/dsp8049.xsd"
2802 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2803 xmlns="http://schemas.dmtf.org/ovf/networkportprofile/1"
2804 xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
2805 xmlns:epasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2806 schema/2/CIM_EthernetPortAllocationSettingData">
2807   <Item>
2808     <epasd:AllowedPriorities>0</epasd:AllowedPriorities>
2809     <epasd:AllowedPriorities>1</epasd:AllowedPriorities>
2810     <epasd:DefaultPriority>0</epasd:DefaultPriority>
2811     <epasd:ElementName>Network Port Profile 2</epasd:ElementName>
2812     <epasd:InstanceID>2</epasd:InstanceID>
2813     <epasd:NetworkPortProfileID>aaaaaaaa-bbbb-cccc-dddd-
2814 ffffffff</epasd:NetworkPortProfileID>
2815     <epasd:NetworkPortProfileIDType>3</epasd:NetworkPortProfileIDType>
2816   </Item>
2817 </NetworkPortProfile>

```

ANNEX F (informative)

2818
2819
2820
2821

Deployment Considerations

2822 This standard defines an OVF package and the main clauses in this standard deal with this subject
2823 matter. However, there are deployment considerations necessary to meet the expectations of the OVF
2824 package author. These are listed below.

F.1 OVF Package Structure Deployment Considerations

2826 A deployment function shall verify the ovf package signature and should validate the certificate.

F.2 Virtual Hardware Deployment Considerations

2828 If there are multiple virtual hardware sections then the deployment function should select the most
2829 appropriate one for the target virtualization platform.

2830 If no backing is specified for a device that requires a backing, the deployment function shall make an
2831 appropriate choice, for example, by prompting the user. More than one backing for a device shall not be
2832 specified.

2833 The deployment function should select the normal value for a resource allocation but may adjust it within
2834 the specified range. The virtualization management may further alter the resource allocation within the
2835 specified range for performance tuning.

F.3 Core Metadata Sections Deployment Considerations

2837 The sharing of disk blocks at runtime is optional and virtualization platform specific and shall not be visible
2838 to the guest software.

2839 A virtualization platform may share storage extents to minimize the amount of space required to support
2840 the virtual systems. If storage extents are shared by the virtualization platform this sharing is not visible to
2841 the guest software.

2842 If present the AnnotationSection element may be displayed during deployment of the OVF package.

2843 If present, the EULASection(s) shall be displayed and accepted during deployment of an OVF package.
2844 If automated deployment is used then the deployment function shall have a methodology to provide
2845 implicit acceptance.

2846 If virtual disks or other files are included by reference the deployment function shall acquire those files
2847 prior to the virtual system being launched.

2848 If the specified boot source is a device type, then the deployment function should try all the devices of
2849 that device type specified.

ANNEX G (informative)

Bibliography

- 2850
2851
2852
2853
- 2854 ISO 9660, *Joliet Extensions Specification*, May 1995,
2855 <http://littlesvr.ca/isomaster/resources/JolietSpecification.html> W3C, *Best Practices for XML*
2856 *Internationalization*, October 2008,
2857 <http://www.w3.org/TR/2008/NOTE-xml-i18n-bp-20080213/>
- 2858 DMTF DSP1044, *Processor Device Resource Virtualization Profile 1.0*
2859 http://www.dmtf.org/standards/published_documents/DSP1044_1.0.pdf
- 2860 DMTF DSP1045, *Memory Resource Virtualization Profile 1.0*
2861 http://www.dmtf.org/standards/published_documents/DSP1045_1.0.pdf
- 2862 DMTF DSP1047, *Storage Resource Virtualization Profile 1.0*
2863 http://www.dmtf.org/standards/published_documents/DSP1047_1.0.pdf
- 2864 DMTF DSP1022, *CPU Profile 1.0*,
2865 http://www.dmtf.org/standards/published_documents/DSP1022_1.0.pdf
- 2866 DMTF DSP1026, *System Memory Profile 1.0*,
2867 http://www.dmtf.org/standards/published_documents/DSP1026_1.0.pdf
- 2868 DMTF DSP1014, *Ethernet Port Profile 1.0*,
2869 http://www.dmtf.org/standards/published_documents/DSP1014_1.0.pdf
- 2870 DMTF DSP1050, *Ethernet Port Resource Virtualization Profile 1.1*
2871 http://www.dmtf.org/standards/published_documents/DSP1050_1.1.pdf
- 2872 DMTF DSP8049, *Network Port Profile XML Schema 1.0*
2873 http://schema.dmtf.org/ovf/networkportprofile/1/DSP8049_1.0.xsd
2874

ANNEX H (informative)

Change Log

2875
2876
2877
2878

Version	Date	Description
1.0.0	2009-02-22	DMTF Standard release
1.1.0	2010-01-12	DMTF Standard release
2.0.0a	2011-06-28	Work in Progress release
2.0.0b	2011-12-14	Work in Progress release
2.0.0c	2012-05-24	Work in Progress release
2.0.0d	2012-09-20	Work in Progress release
2.0.0	2012-10-29	DMTF Standard release
2.1.0	2013-01-10	Start of 2.1 specification development Larry's comments and notes.
2.1.0	2013-01-14	wgv 0.5.2 – results of editing session
2.1.0	2013-01-21	wgv 0.6.0 – added deployment annex
2.1.0	2013-01-22	wgv 0.6.1 – added example annex
2.1.0	2013-01-23	wgv 0.6.2 – added references to examples
2.1.0	2013-02-12	wgv 0.6.3 – clean copy changes accepted
2.1.0	2013-02-19	wgv 0.6.4 – based on 0.6.2 with Jeff's comments added
2.1.0	2013-02-20	wgv 0.6.5 – deployment function & virtualization platform
2.1.0	2013-03-06	wgv 0.6.6 – interim during work group f2f
2.1.0	2013-03-11	wgv 0.6.7 – output of work group f2f. format changes accepted, example and deployment changes accepted. Wording changes are marked and several comments need some further discussion
2.1.0	2013-03-28	wgv 0.6.8 – updated sub-clause 9.11 and clause 11 per Ali/Wheeler proposal
2.1.0	2013-04-05	wgv 0.6.9 – update Annex F on deployment
2.1.0	2013-05-02	wgv 0.6.10 – RFC comment resolution
2.1.0a	2013-05-29	work in progress release

2879