



1
2
3
4

Document Number: DSP0243

Date: 2012-12-13

Version: 2.0.0

5 **Open Virtualization Format Specification**

6 **Document Type: Specification**

7 **Document Status: DMTF Standard**

8 **Document Language: en-US**

9 Copyright notice

10 Copyright © 2012 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25 withdrawn or modified after publication, and shall be indemnified and held harmless by any party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 Packages	12
39	5.1 OVF Package Structure.....	12
40	5.2 Virtual Disk Formats.....	14
41	5.3 Distribution as a Single File	14
42	5.4 Distribution as a Set of Files	15
43	6 OVF Descriptor.....	15
44	7 Envelope Element	16
45	7.1 File References	17
46	7.2 Content Element	18
47	7.3 Extensibility	19
48	7.4 Conformance	20
49	8 Virtual Hardware Description.....	20
50	8.1 VirtualHardwareSection	20
51	8.2 Extensibility	23
52	8.3 Virtual Hardware Elements	23
53	8.4 Ranges on Elements.....	25
54	9 Core Metadata Sections in version 2	27
55	9.1 DiskSection	28
56	9.2 NetworkSection.....	29
57	9.3 ResourceAllocationSection	30
58	9.4 AnnotationSection.....	30
59	9.5 ProductSection.....	31
60	9.6 EulaSection.....	34
61	9.7 StartupSection	34
62	9.8 DeploymentOptionSection	35
63	9.9 OperatingSystemSection	37
64	9.10 InstallSection.....	38
65	9.11 EnvironmentFilesSection	38
66	9.12 BootDeviceSection.....	39
67	9.13 SharedDiskSection	40
68	9.14 ScaleOutSection	41
69	9.15 PlacementGroupSection and PlacementSection.....	42
70	9.16 Encryption Section	44
71	10 Internationalization	46
72	10.1 Internal Resource Bundles	46
73	10.2 External Resource Bundles	47
74	10.3 Message Content in External File.....	47
75	11 OVF Environment.....	48
76	11.1 Environment Document	48
77	11.2 Transport.....	49
78	ANNEX A (informative) Symbols and Conventions	51
79	ANNEX B (normative) OVF XSD	52
80	ANNEX C (informative) OVF Mime Type Registration Template	53
81	ANNEX D (informative) Network Port Profile Examples	55

82	D.1	Example 1 (OVF Descriptor for One Virtual System and One Network with an Inlined Network Port Profile).....	55
83			
84	D.2	Example 2 (OVF Descriptor for One Virtual System and One Network with a Locally Referenced Network Port Profile).....	57
85			
86	D.3	Example 3 (OVF Descriptor for One Virtual System and One Network with a Network Port Profile referenced by a URI).....	58
87			
88	D.4	Example 4 (OVF Descriptor for Two Virtual Systems and One Network with Two Network Port Profiles referenced by URIs).....	60
89			
90	D.5	Example 5 (networkportprofile1.xml).....	63
91	D.6	Example 6 (networkportprofile2.xml).....	63
92	ANNEX E (informative) Change Log.....		64
93	Bibliography		65
94			

95 **Tables**

96 Table 1 – XML Namespace Prefixes 16

97 Table 2 – Actions for Child Elements with `ovf:required` Attribute..... 23

98 Table 3 – HostResource Element 24

99 Table 4 – Elements for Virtual Devices and Controllers 25

100 Table 5 – Core Metadata Sections in version 2 27

101 Table 6 – Property Types 33

102 Table 7 – Property Qualifiers 34

103 Table 8 – Core Sections..... 49

104

105

Foreword

106 The *Open Virtualization Format Specification* (DSP0243) was prepared by the System Virtualization,
107 Partitioning, and Clustering Working Group of the DMTF.

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

110

111	Lawrence Lamers	VMware Inc. (Chair)
112	Hemal Shah	Broadcom Corporation (co-Editor)
113	Steffen Grarup	VMware Inc. (co-Editor)
114		
115	Vincent Kowalski	BMC Software
116	Hemal Shah	Broadcom Corporation
117	John Crandall	Brocade Communications Systems
118	Marvin Waschke	CA Technologies
119	Naveen Joy	Cisco
120	Steven Neely	Cisco
121	Shishir Pardikar	Citrix Systems Inc.
122	Thomas Root	Citrix Systems Inc.
123	Richard Landau	DMTF Fellow
124	Jacques Durand	Fujitsu
125	Derek Coleman	Hewlett-Packard Company
126	Robert Freund	Hitachi, Ltd.
127	Fred Maciel	Hitachi, Ltd.
128	Eric Wells	Hitachi, Ltd.
129	Abdellatif Touimi	Huawei
130	Jeff Wheeler	Huawei
131	HengLiang Zhang	Huawei
132	Oliver Benke	IBM
133	Ron Doyle	IBM
134	Michael Gering	IBM
135	Michael Johanssen	IBM
136	Andreas Maier	IBM
137	Marc-Arthur Pierre-Louis	IBM
138	John Leung	Intel Corporation
139	Nitin Bhat	Microsoft Corporation
140	Maurizio Carta	Microsoft Corporation
141	Monica Martin	Microsoft Corporation
142	John Parchem	Microsoft Corporation
143	Ed Reed	Microsoft Corporation
144	Nihar Shah	Microsoft Corporation
145	Cheng Wei	Microsoft Corporation
146	Narayan Venkat	NetApp
147	Tatyana Bagerman	Oracle
148	Srinivas Maturi	Oracle
149	Dr. Fermín Galán Márquez	Telefónica
150	Miguel Ángel Peñalvo	Telefónica
151	Dr. Fernando de la Iglesia	Telefónica
152	Álvaro Polo	Telefónica
153	Steffen Grarup	VMware Inc.
154	Lawrence Lamers	VMware Inc.
155	Rene Schmidt	VMware Inc.
156	Paul Ferdinand	WBEM Solutions

157	Junsheng Chu	ZTE Corporation
158	Bhumip Khasnabish	ZTE Corporation
159	Ghazanfar Ali	ZTE Corporation

160

Introduction

161 The *Open Virtualization Format (OVF) Specification* describes an open, secure, portable, efficient and
162 extensible format for the packaging and distribution of software to be run in virtual machines. The key
163 properties of the format are as follows:

164 • **Optimized for distribution**

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

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

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

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

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

175 • **Portable VM packaging**

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

180 • **Vendor and platform independent**

181 OVF does not rely on the use of a specific host platform, virtualization platform, or guest
182 operating system.

183 • **Extensible**

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

187 • **Localizable**

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

191 • **Open standard**

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

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

196

Open Virtualization Format Specification

197 1 Scope

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

200 This version of the specification (2.0) is intended to allow OVF 1.x tools to work with OVF 2.0 descriptors
201 in the following sense:
202

- 203 • Existing OVF 1.x tools should be able to parse OVF 2.0 descriptors.
- 204 • Existing OVF 1.x tools should be able to give warnings/errors if dependencies to 2.0 features are
205 required for correct operation.

206 2 Normative References

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

210 [ISO/IEC/IEEE 9945:2009](http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50516): Information technology -- Portable Operating System Interface (POSIX®) Base
211 Specifications, Issue 7
212 http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50516

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

215 DMTF DSP0230, *WS-CIM Mapping Specification 1.1*,
216 http://www.dmtf.org/standards/published_documents/DSP0230_1.1.pdf

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

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

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

223 DMTF DSP8023, *Open Virtualization Format (OVF) 2 XML Schema*,
224 <http://schemas.dmtf.org/ovf/envelope/2/dsp8023.xsd>

225 DMTF DSP8049, *Network Port Profile XML Schema*,
226 <http://schemas.dmtf.org/ovf/networkportprofile/1/dsp8049.xsd>

227 IETF RFC1738, T. Berners-Lee, *Uniform Resource Locators (URL)*, December 1994,
228 <http://tools.ietf.org/html/rfc1738>

229 IETF RFC1952, P. Deutsch, *GZIP file format specification version 4.3*, May 1996,
230 <http://tools.ietf.org/html/rfc1952>

231 IETF Standard 68, *Augmented BNF for Syntax Specifications: ABNF*,
232 <http://tools.ietf.org/html/rfc5234>

- 233 IETF RFC2616, R. Fielding et al, *Hypertext Transfer Protocol – HTTP/1.1*, June 1999,
234 <http://tools.ietf.org/html/rfc2616>
- 235 IETF Standard 66, *Uniform Resource Identifiers (URI): Generic Syntax*,
236 <http://tools.ietf.org/html/rfc3986>
- 237 ISO 9660, 1988 Information processing-Volume and file structure of CD-ROM for information interchange,
238 http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=17505
- 239 ISO, ISO/IEC Directives, Part 2, *Rules for the structure and drafting of International Standards*,
240 <http://isotc.iso.org/livelink/livelink.exe?func=ll&objId=4230456&objAction=browse&sort=subtype>
- 241 W3C, *XML Schema Part 1: Structures Second Edition*, 28 October 2004. W3C Recommendation. URL:
242 <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/>
- 243 W3C, *XML Schema Part 2: Datatypes Second Edition*, 28 October 2004. W3C Recommendation. URL:
244 <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/>
- 245 XML Encryption Syntax and Processing Version 1.1, 13 March 2012, W3C Candidate Recommendation
246 <http://www.w3.org/TR/2012/CR-xmlenc-core1-20120313/>
- 247 FIPS 180-2: Secure Hash Standard (SHS)
248 <http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf>

249 3 Terms and Definitions

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

251 3.1

252 **can**

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

254 3.2

255 **cannot**

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

257 3.3

258 **conditional**

259 indicates requirements to be followed strictly to conform to the document when the specified conditions
260 are met

261 3.4

262 **mandatory**

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

265 3.5

266 **may**

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

268 3.6

269 **need not**

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

271 3.7

- 272 **optional**
273 indicates a course of action permissible within the limits of the document
- 274 **3.8**
275 **shall**
276 indicates requirements to be followed strictly to conform to the document and from which no deviation is
277 permitted
- 278 **3.9**
279 **shall not**
280 indicates requirements to be followed strictly to conform to the document and from which no deviation is
281 permitted
- 282 **3.10**
283 **should**
284 indicates that among several possibilities, one is recommended as particularly suitable, without
285 mentioning or excluding others, or that a certain course of action is preferred but not necessarily required
- 286 **3.11**
287 **should not**
288 indicates that a certain possibility or course of action is deprecated but not prohibited
- 289 **3.12**
290 **appliance**
291 see [virtual appliance](#)
- 292 **3.13**
293 **deployment platform**
294 the product that installs an OVF package
- 295 **3.14**
296 **guest software**
297 the software that runs inside a virtual machine
298 The guest is typically an operating system and some user-level applications and services.
- 299 **3.15**
300 **OVF package**
301 OVF XML descriptor file accompanied by zero or more files
- 302 **3.16**
303 **OVF descriptor**
304 OVF XML descriptor file
- 305 **3.17**
306 **platform**
307 see [deployment platform](#)
- 308 **3.18**
309 **virtual appliance**
310 a service delivered as a complete software stack installed on one or more virtual machines
311 A virtual appliance is typically expected to be delivered in an OVF package.

312 **3.19**
313 **virtual hardware**
314 the processor, memory and I/O resources of a virtual computer system

315 **3.20**
316 **virtual machine**
317 as defined in System Virtualization Profile

318 **3.21**
319 **virtual machine collection**
320 a collection comprised of a set of virtual machines. This service component can be a:
321 - simple set of one or more virtual machines, or
322 - a complex service component built out of a combination of virtual machines and other virtual
323 machine collections that enables nested complex service components.

324 **4 Symbols and Abbreviated Terms**

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

326 **4.1.1**
327 **CIM**
328 Common Information Model

329 **4.1.2**
330 **IP**
331 Internet Protocol

332 **4.1.3**
333 **OVF**
334 Open Virtualization Format

335 **4.1.4**
336 **VM**
337 Virtual Machine

338 **5 OVF Packages**

339 **5.1 OVF Package Structure**

340 An OVF package shall consist of the following files:

- 341 • one OVF descriptor with extension `.ovf`
- 342 • zero or one OVF manifest with extension `.mf`
- 343 • zero or one OVF certificate with extension `.cert`
- 344 • zero or more disk image files
- 345 • zero or more additional resource files, such as ISO images

346 The file extensions `.ovf`, `.mf` and `.cert` shall be used.

347 EXAMPLE 1: The following list of files is an example of an OVF package:

```
348 package.ovf
349 package.mf
350 de-DE-resources.xml
351 vmdisk1.vmdk
352 vmdisk2.vhd
353 resource.iso
```

354 An OVF package can be stored as either a single unit or a set of files, as described in 5.3 and 5.4. Both
355 modes shall be supported.

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

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

364 The format of the manifest file is as follows:

```
365 manifest_file = *( file_digest )
366 file_digest  = algorithm "(" file_name ")" "=" sp digest nl
367 algorithm    = "SHA1" | "SHA256"
368 digest       = *( hex-digit )
369 hex-digit    = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" | "a" |
370 "b" | "c" | "d" | "e" | "f"
371 sp           = %x20
372 nl           = %x0A
```

373 EXAMPLE 2: The following example show the partial contents of a manifest file:

```
374 SHA256(package.ovf)= 9902cc5ec4f4a00cabbff7b60d039263587ab430d5fdbc5cd5e8707391c90a4
375 SHA256(vmdisk.vmdk)= aab66c4d70e17cec2236a651a3fc618cafc5ec6424122904dc0b9c286fce40c2
```

376 An OVF package may be signed by signing the manifest file. The digest of the manifest file is stored in a
377 certificate file with extension `.cert` file along with the base64-encoded X.509 certificate. The `.cert` file
378 shall have the same base name as the `.ovf` file and be a sibling of the `.ovf` file. A consumer of the OVF
379 package shall verify the signature and should validate the certificate. The format of the certificate file shall
380 be as follows:

```
381 certificate_file = manifest_digest certificate_part
382 manifest_digest = algorithm "(" file_name ")" "=" sp signed_digest nl
383 algorithm       = "SHA1" | "SHA256"
384 signed_digest   = *( hex-digit)
385 certificate_part = certificate_header certificate_body certificate_footer
386 certificate_header = "-----BEGIN CERTIFICATE-----" nl
387 certificate_footer = "-----END CERTIFICATE-----" nl
388 certificate_body  = base64-encoded-certificate nl
389                  ; base64-encoded-certificate is a base64-encoded X.509
390                  ; certificate, which may be split across multiple lines
391 hex-digit        = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" | "a"
392 | "b" | "c" | "d" | "e" | "f"
393 sp               = %x20
394 nl               = %x0A
```

395 EXAMPLE 3: The following list of files is an example of a signed OVF package:

```
396 package.ovf
397 package.mf
398 package.cert
399 de-DE-resources.xml
400 vmdisk1.vmdk
401 vmdisk2.vmdk
402 resource.iso
```

403 EXAMPLE 4: The following example shows the contents of a sample OVF certification file, where the SHA1 digest
404 of the manifest file has been signed with a 512 bit key:

```
405 SHA1(package.mf) = 7f4b8efb8fe20c06df1db68281a63f1b088e19dbf00e5af9db5e8e3e319de
406 7019db88a3bc699bab6ccd9e09171e21e88ee20b5255cec3fc28350613b2c529089
407 -----BEGIN CERTIFICATE-----
408 MIIBGjCCASwCAQQwDQYJKoZIhvcNAQEEBQAwoDELMakGA1UEBhMCQVUxDDAKBgNV
409 BAgtA1FMRDEbMBkGA1UEAxMSU1NMZWV5L3JzYSB0ZXN0IENBMB4XDtk1MTAwOTIz
410 MzIwNVVoXDTk4MDcwNTIzMzIwNVVowYDELMAkGA1UEBhMCQVUxDDAKBgNVBAgtA1FM
411 RDEZMBcGA1UEChMQTWluY29tIFB0eS4gTHRkLjELMAkGA1UECxmCQ1MxGzAZBgNV
412 BAMTElNTTGVheSBkZW1vIHNLcnZlcjBcMA0GCsGSIb3DQEBAQUAA0sAMEgCQQC3
413 LCXcScWua0PFLkHBLm2VejqpA1F4RQ8q0VjRiPafjx/Z/aWH3ipdMVvuJGa/wFXb
414 /nDFLDlFwP+oCPwhBtVPAGMBAAEwDQYJKoZIhvcNAQEEBQADQQARNFsihWIjBzb0
415 DcsU0BvL2bvSwJrPEqFlkDq3F4M6EgutL9axEcANWgbbEdAvNJD1dmEmoWny27Pn
416 Ims6ZOZB
417 -----END CERTIFICATE-----
```

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

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

425 5.2 Virtual Disk Formats

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

432 5.3 Distribution as a Single File

433 An OVF package may be stored as a single file using the TAR format. The extension of that file shall be
434 `.ova` (open virtual appliance or application).

435 EXAMPLE: The following example shows a sample filename for an OVF package of this type:

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

437 For OVF packages stored as single file, all file references in the OVF descriptor shall be relative-path
438 references and shall point to files included in the TAR archive. Relative directories inside the archive are
439 allowed, but relative-path references shall not contain “..” dot-segments.

440 Ordinarily, a TAR extraction tool would have to scan the whole archive, even if the file requested is found
441 at the beginning, because replacement files can be appended without modifying the rest of the archive.
442 Entries in an OVF TAR file shall exist only once.

443 In addition, the entries shall be in one of the following orders inside the archive:

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

448 1) OVF descriptor
449 2) OVF manifest
450 3) OVF certificate
451 4) The remaining files shall be in the same order as listed in the References section (see 7.1).
452 Note that any external string resource bundle files for internationalization shall be first in the
453 References section (see clause 10).

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

460 For deployment, the ordering restriction ensures that it is possible to extract the OVF descriptor from an
461 OVF TAR file without scanning the entire archive. For generation, the ordering restriction ensures that an
462 OVF TAR file can easily be generated on-the-fly. The restrictions do not prevent OVF TAR files from
463 being created using standard TAR packaging tools.

464 The TAR format used shall comply with the USTAR (Uniform Standard Tape Archive) format as defined
465 by the [ISO/IEC/IEEE 9945:2009](#).

466 5.4 Distribution as a Set of Files

467 An OVF package can be made available as a set of files, for example on a standard Web server.

468 EXAMPLE: An example of an OVF package as a set of files on Web server follows:

```
469 http://mywebsite/virtualappliances/package.ovf  
470 http://mywebsite/virtualappliances/vmdisk1.vmdk  
471 http://mywebsite/virtualappliances/vmdisk2.vmdk  
472 http://mywebsite/virtualappliances/resource.iso  
473 http://mywebsite/virtualappliances/de-DE-resources.xml
```

474 6 OVF Descriptor

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

478 The DMTF DSP8023 schema definition file for the OVF descriptor contains the elements and attributes.
479 The OVF descriptor shall validate with the DMTF [DSP8023](#).

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

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

485

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

486 7 Envelope Element

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

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

- 490 • A version indication, defined by the XML namespace URIs.
- 491 • A list of file references to all external files that are part of the OVF package, defined by the
 492 `References` element and its `File` child elements. These are typically virtual disk files, ISO
 493 images, and internationalization resources.
- 494 • A metadata part, defined by section elements, as defined in clause 9.
- 495 • A description of the content, either a single virtual machine (`VirtualSystem` element) or a
 496 collection of multiple virtual machines (`VirtualSystemCollection` element).
- 497 • A specification of message resource bundles for zero or more locales, defined by a `Strings`
 498 element for each locale.

499 **EXAMPLE:** An example of the structure of an OVF descriptor with the top-level `Envelope` element follows:

```

500 <?xml version="1.0" encoding="UTF-8"?>
501 <Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
502   xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-
503   schema/2/CIM_VirtualSystemSettingData"
504   xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
505   schema/2/CIM_ResourceAllocationSettingData"
506   xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/2"
507   xmlns="http://schemas.dmtf.org/ovf/envelope/2"
508   xml:lang="en-US">
509   <References>
510     <File ovf:id="de-DE-resources.xml" ovf:size="15240"
511       ovf:href="http://mywebsite/virtualappliances/de-DE-resources.xml"/>
512     <File ovf:id="file1" ovf:href="vmdisk1.vmdk" ovf:size="180114671"/>
513     <File ovf:id="file2" ovf:href="vmdisk2.vmdk" ovf:size="4882023564"
514   ovf:chunkSize="2147483648"/>
515     <File ovf:id="file3" ovf:href="resource.iso" ovf:size="212148764"
516   ovf:compression="gzip"/>
517     <File ovf:id="icon" ovf:href="icon.png" ovf:size="1360"/>
518   </References>
519   <!-- Describes meta-information about all virtual disks in the package -->
520   <DiskSection>
521     <Info>Describes the set of virtual disks</Info>
  
```



```

522     <!-- Additional section content -->
523     </DiskSection>
524     <!-- Describes all networks used in the package -->
525     <NetworkSection>
526         <Info>List of logical networks used in the package</Info>
527         <!-- Additional section content -->
528     </NetworkSection>
529     <SomeSection ovf:required="false">
530         <Info>A plain-text description of the content</Info>
531         <!-- Additional section content -->
532     </SomeSection>
533     <!-- Additional sections can follow -->
534     <VirtualSystemCollection ovf:id="Some Product">
535         <!-- Additional sections including VirtualSystem or VirtualSystemCollection-->
536     </VirtualSystemCollection >
537     <Strings xml:lang="de-DE">
538         <!-- Specification of message resource bundles for de-DE locale -->
539     </Strings>
540 </Envelope>

```

541 The optional `xml:lang` attribute on the `Envelope` element shall specify the default locale for messages
542 in the descriptor. The optional `Strings` elements shall contain string resource bundles for different
543 locales. See clause 10 for more details on internationalization support.

544 7.1 File References

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

548 External string resource bundle files for internationalization shall be placed first in the `References`
549 element, see clause 10 for details.

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

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

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

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

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

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

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

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

583 If the OVF package has a manifest file, the file name in the manifest entries shall match the value of the
 584 `ovf:href` attribute for the file, except if the file is split into multiple chunks, in which case the `chunk-`
 585 `url` shall be used, and the manifest file shall contain an entry for each individual chunk. If chunked files
 586 are used, the manifest file may contain an entry for the entire file; and if present this digest shall also be
 587 verified.

588 EXAMPLE 1: The following example shows different types of file references:

```
589 <File ovf:id="disk1" ovf:href="disk1.vmdk"/>
590 <File ovf:id="disk2" ovf:href="disk2.vmdk" ovf:size="5368709120"
591         ovf:chunkSize="2147483648"/>
592 <File ovf:id="iso1" ovf:href="resources/image1.iso"/>
593 <File ovf:id="iso2" ovf:href="http://mywebsite/resources/image2.iso"/>
```

594 EXAMPLE 2: The following example shows manifest entries corresponding to the file references above:

```
595 SHA1(disk1.vmdk) = 3e19644ec2e806f38951789c76f43e4a0ec7e233
596 SHA1(disk2.vmdk.000000000) = 4f7158731ff434380bf217da248d47a2478e79d8
597 SHA1(disk2.vmdk.000000001) = 12849daeeaf43e7a89550384d26bd437bb8defaf
598 SHA1(disk2.vmdk.000000002) = 4cdd21424bd9eeafa4c42112876217de2ee5556d
599 SHA1(resources/image1.iso) = 72b37ff3fdd09f2a93f1b8395654649b6d06b5b3
600 SHA1(http://mywebsite/resources/image2.iso) =
601 d3c2d179011c970615c5cf10b30957d1c4c968ad
```

602 7.2 Content Element

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

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

609 The `VirtualSystem` element describes a single virtual machine and is simply a container of section
 610 elements. These section elements describe virtual hardware, resources, and product information and are
 611 described in detail in clauses 8 and 9.

612 An example of a `VirtualSystem` element structure is as follows:

```
613 <VirtualSystem ovf:id="simple-app">
614   <Info>A virtual machine</Info>
615   <Name>Simple Appliance</Name>
616   <SomeSection>
617     <!-- Additional section content -->
618   </SomeSection>
619   <!-- Additional sections can follow -->
```

620 `</VirtualSystem>`

621 The `VirtualSystemCollection` element is a container of multiple `VirtualSystem` or
 622 `VirtualSystemCollection` elements. Thus, arbitrary complex configurations can be described. The
 623 section elements at the `VirtualSystemCollection` level describe appliance information, properties,
 624 resource requirements, and so on, and are described in detail in clause 9.

625 An example of a `VirtualSystemCollection` element structure is as follows:

```
626 <VirtualSystemCollection ovf:id="multi-tier-app">
627   <Info>A collection of virtual machines</Info>
628   <Name>Multi-tiered Appliance</Name>
629   <SomeSection>
630     <!-- Additional section content -->
631   </SomeSection>
632   <!-- Additional sections can follow -->
633   <VirtualSystem ovf:id="...">
634     <!-- Additional sections -->
635   </VirtualSystem>
636   <!-- Additional VirtualSystem or VirtualSystemCollection elements can follow-->
637 </VirtualSystemCollection>
```

638 All elements in the `Content` substitution group contain an `Info` element and may contain a `Name`
 639 element. The `Info` element contains a human readable description of the meaning of this entity. The
 640 `Name` element is an optional localizable display name of the content. See clause 10 for details on how to
 641 localize the `Info` and `Name` element.

642 7.3 Extensibility

643 This specification allows custom meta-data to be added to OVF descriptors in several ways:

- 644 • New section elements may be defined as part of the `Section` substitution group, and used
 645 where the OVF schemas allow sections to be present. All subtypes of `Section` contain an `Info`
 646 element that contains a human readable description of the meaning of this entity. The values of
 647 `Info` elements can be used, for example, to give meaningful warnings to users when a section is
 648 being skipped, even if the parser does not know anything about the section. See clause 10 for
 649 details on how to localize the `Info` element.
- 650 • The OVF schemas use an open content model, where all existing types may be extended at the
 651 end with additional elements. Extension points are declared in the OVF schemas with `xs:any`
 652 declarations with `namespace="##other"`.
- 653 • The OVF schemas allow additional attributes on existing types.

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

656 On custom elements, a Boolean `ovf:required` attribute specifies whether the information in the
 657 element is required for correct behavior or optional. If not specified, the `ovf:required` attribute defaults
 658 to TRUE. A consumer of an OVF package that detects an extension that is required and that it does not
 659 understand shall fail.

660 For known `Section` elements, if additional child elements that are not understood are found and the
 661 value of their `ovf:required` attribute is TRUE, the consumer of the OVF package shall interpret the
 662 entire section as one it does not understand. The check is not recursive; it applies only to the direct
 663 children of the `Section` element. This behavior ensures that older parsers reject newer OVF
 664 specifications, unless explicitly instructed not to do so.

665 On custom attributes, the information in the attribute shall not be required for correct behavior.

666 EXAMPLE 1:

```
667 <!-- Optional custom section example -->
668 <otherns:IncidentTrackingSection ovf:required="false">
669   <Info>Specifies information useful for incident tracking purposes</Info>
670   <BuildSystem>Acme Corporation Official Build System</BuildSystem>
671   <BuildNumber>102876</BuildNumber>
672   <BuildDate>10-10-2008</BuildDate>
673 </otherns:IncidentTrackingSection>
```

674 EXAMPLE 2:

```
675 <!-- Open content example (extension of existing type) -->
676 <AnnotationSection>
677   <Info>Specifies an annotation for this virtual machine</Info>
678   <Annotation>This is an example of how a future element (Author) can still be
679     parsed by older clients</Annotation>
680   <!-- AnnotationSection extended with Author element -->
681   <otherns:Author ovf:required="false">John Smith</otherns:Author>
682 </AnnotationSection>
```

683 EXAMPLE 3:

```
684 <!-- Optional custom attribute example -->
685 <Network ovf:name="VM network" otherns:desiredCapacity="1 Gbit/s">
686   <Description>The main network for VMs</Description>
687 </Network>
```

688 7.4 Conformance

689 This specification defines three conformance levels for OVF descriptors, with 1 being the highest level of
690 conformance:

- 691 • OVF descriptor uses only sections and elements and attributes that are defined in this
692 specification.
693 Conformance Level: 1.
- 694 • OVF descriptor uses custom sections or elements or attributes that are not defined in this
695 specification, and all such extensions are optional as defined in 7.3.
696 Conformance Level: 2.
- 697 • OVF descriptor uses custom sections or elements that are not defined in this specification and at
698 least one such extension is required as defined in 7.3. The definition of all required extensions
699 shall be publicly available in an open and unencumbered XML Schema. The complete
700 specification may be inclusive in the XML schema or available as a separate document.
701 Conformance Level: 3.

702 The use of conformance level 3 limits portability and should be avoided if at all possible.

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

705 8 Virtual Hardware Description

706 8.1 VirtualHardwareSection

707 Each VirtualSystem element may contain one or more VirtualHardwareSection elements, each of which
708 describes the virtual hardware required by the virtual system. The virtual hardware required by a virtual
709 machine is specified in VirtualHardwareSection elements. This specification supports abstract or
710 incomplete hardware descriptions in which only the major devices are described. The virtualization

711 platform may create additional virtual hardware controllers and devices, as long as the required devices
712 listed in the descriptor are realized.

713
714 This virtual hardware description is based on the CIM classes `CIM_VirtualSystemSettingData`,
715 `CIM_ResourceAllocationSettingData`, `CIM_EthernetPortAllocationSettingData`, and
716 `CIM_StorageAllocationSettingData`. The XML representation of the CIM model is based on the
717 WS-CIM mapping (DSP0230). Note: This means that the XML elements that belong to the class
718 complex type should be ordered by Unicode code point (binary) order of their CIM property name
719 identifiers.

720 EXAMPLE: Example of `VirtualHardwareSection`:

```

721 <VirtualHardwareSection>
722   <Info>Memory = 4 GB, CPU = 1 GHz, Disk = 100 GB, 1 Ethernet nic</Info>
723   <Item>
724     <rasd:AllocationUnits>Hertz*10^9</rasd:AllocationUnits>
725     <rasd:Description>Virtual CPU</rasd:Description>
726     <rasd:ElementName>1 GHz virtual CPU</rasd:ElementName>
727     <rasd:InstanceID>1</rasd:InstanceID>
728     <rasd:Reservation>1</rasd:Reservation>
729     <rasd:ResourceType>3</rasd:ResourceType>
730     <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
731     <rasd:VirtualQuantityUnit>Count</ rasd:VirtualQuantityUnit>
732   </Item>
733   <Item>
734     <rasd:AllocationUnits>byte*2^30</rasd:AllocationUnits>
735     <rasd:Description>Memory</rasd:Description>
736     <rasd:ElementName>1 GByte of memory</rasd:ElementName>
737     <rasd:InstanceID>2</rasd:InstanceID>
738     <rasd:Limit>4</rasd:Limit>
739     <rasd:Reservation>4</rasd:Reservation>
740     <rasd:ResourceType>4</rasd:ResourceType>
741   </Item>
742   <EthernetPortItem>
743     <epasd:Address>00-16-8B-DB-00-5E</epasd:Address>
744     <rasd:AllocationUnits>bit / second *2^30 </rasd:AllocationUnits> VERIFY
745 the PUnit for Gbits per second
746     <epasd:Connection>VM Network</epasd:Connection>
747     <epasd:Description>Virtual NIC</epasd:Description>
748
749     <epasd:ElementName>Ethernet Port</epasd:ElementName>
750     <epasd:InstanceID>3</epasd:InstanceID>
751     <epasd:NetworkPortProfileID>1</epasd:NetworkPortProfileID>
752     <epasd:NetworkPortProfileIDType>4</epasd:NetworkPortProfileIDType>
753     <epasd:ResourceType>10</epasd:ResourceType>
754     <epasd:VirtualQuantity>1</epasd:VirtualQuantity>
755     <epasd:VirtualQuantityUnits>Count</epasd:VirtualQuantityUnits>
756   </EthernetPortItem>
757   <StorageItem>
758     <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
759     <sasd:Description>Virtual Disk</sasd:Description>
760     <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
761     <sasd:InstanceID>4</sasd:InstanceID>
762     <sasd:Reservation>100</sasd:Reservation>
763     <sasd:ResourceType>31</sasd:ResourceType>
764     <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
765     <sasd:VirtualQuantityUnit>Count</sasd:VirtualQuantityUnit>
766   </StorageItem>
767 </VirtualHardwareSection>

```

- 768 A `VirtualSystem` element shall have a `VirtualHardwareSection` direct child element.
769 `VirtualHardwareSection` shall not be a direct child element of a `VirtualSystemCollection`
770 element and of an `Envelope` element.
- 771 Multiple `VirtualHardwareSection` element occurrences are allowed within a single `VirtualSystem`
772 element. The consumer of the OVF package should select the most appropriate virtual hardware
773 description for the particular virtualization platform. A `VirtualHardwareSection` element may contain
774 an `ovf:id` attribute which can be used to identify the element. If present the attribute value must be
775 unique within the `VirtualSystem`.
- 776 The `ovf:transport` attribute specifies the types of transport mechanisms by which properties are
777 passed to the virtual machine in an OVF environment document. This attribute supports a pluggable and
778 extensible architecture for providing guest/platform communication mechanisms. Several transport types
779 may be specified separated by single space character. See 9.5 for a description of properties and clause
780 11 for a description of transport types and OVF environments.
- 781 A `VirtualHardwareSection` element contains sub elements that describe virtual system and virtual
782 hardware resources (CPU, memory, network, and storage).
- 783 A `VirtualHardwareSection` element shall have zero or one `System` direct child element, followed by
784 zero or more `Item` direct child elements, zero or more `EthernetPortItem` direct child elements, and
785 zero or more `StorageItem` direct child elements.
- 786 The `System` element is an XML representation of the values of one or more properties of the CIM class
787 `CIM_VirtualSystemSettingData`. The `vssd:VirtualSystemType`, a direct child element of
788 `System` element, specifies a virtual system type identifier, which is an implementation defined string that
789 uniquely identifies the type of the virtual system. For example, a virtual system type identifier could be
790 `vmx-4` for VMware's fourth-generation virtual hardware or `xen-3` for Xen's third-generation virtual
791 hardware. Zero or more virtual system type identifiers may be specified separated by single space
792 character. In order for the OVF virtual system to be deployable on a target platform, the virtual machine
793 on the target platform should support at least one of the virtual system types identified in the
794 `vssd:VirtualSystemType` elements. The virtual system type identifiers specified in
795 `vssd:VirtualSystemType` elements are expected to be matched against the values of property
796 `VirtualSystemTypesSupported` of CIM class `CIM_VirtualSystemManagementCapabilities`.
- 797 The virtual hardware characteristics are described as a sequence of `Item` elements. The `Item` element
798 is an XML representation of an instance of the CIM class `CIM_ResourceAllocationSettingData`.
799 The element can describe all memory and CPU requirements as well as virtual hardware devices.
- 800 Multiple device subtypes may be specified in an `Item` element, separated by a single space character.
- 801 EXAMPLE:
802

```
<rasd:ResourceSubType>buslogic lsilogic</rasd:ResourceSubType>
```
- 803 The network hardware characteristics are described as a sequence of `EthernetPortItem` elements.
804 The `EthernetPortItem` element is an XML representation of the values of one or more properties of
805 the CIM class `CIM_EthernetPortAllocationSettingData`.
- 806 The storage hardware characteristics are described as a sequence of `StorageItem` elements. The
807 `StorageItem` element is an XML representation of the values of one or more properties of the CIM class
808 `CIM_StorageAllocationSettingData`.

809 **8.2 Extensibility**

810 The optional `ovf:required` attribute on the `Item`, `EthernetPortItem`, or `StorageItem`
 811 element specifies whether the realization of the element (for example, a CD-ROM or USB controller) is
 812 required for correct behavior of the guest software. If not specified, `ovf:required` defaults to TRUE.

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

816 **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

817 **8.3 Virtual Hardware Elements**

818 The element type of the `Item` element in a `VirtualHardwareSection` element is
 819 `CIM_ResourceAllocationSettingData_Type` as defined in [http://schemas.dmtf.org/wbem/wscim/1/cim-](http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData.xsd)
 820 [schema/2/CIM_ResourceAllocationSettingData.xsd](http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData.xsd).

821 The child elements of `Item` represent the values of one or more properties exposed by the
 822 `CIM_ResourceAllocationSettingData` class. They have the semantics of defined settings as
 823 defined in [DSP1041](#), any profiles derived from [DSP1041](#) for specific resource types, and this document.

824 EXAMPLE: The following example shows a description of memory size:

```
825 <Item>
826   <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
827   <rasd:Description>Memory Size</rasd:Description>
828   <rasd:ElementName>256 MB of memory</rasd:ElementName>
829   <rasd:InstanceID>2</rasd:InstanceID>
830   <rasd:ResourceType>4</rasd:ResourceType>
831   <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
832 </Item>
```

833 The element type of the `EthernetPortItem` element in a `VirtualHardwareSection` element is
 834 `CIM_EthernetPortAllocationSettingData_Type` as defined in [http://schemas.dmtf.org/wbem/wscim/1/cim-](http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_EthernetPortAllocationSettingData.xsd)
 835 [schema/2/CIM_EthernetPortAllocationSettingData.xsd](http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_EthernetPortAllocationSettingData.xsd).

836 The child elements represent the values of one or more properties exposed by the
 837 `CIM_EthernetPortAllocationSettingData` class. They have the semantics of defined settings as
 838 defined in [DSP1050](#), any profiles derived from [DSP1050](#) for specific Ethernet port resource types, and
 839 this document.

840 EXAMPLE: The following example shows a description of a virtual Ethernet adapter:

```
841 <EthernetPortItem>
842   <epasd:Address>00-16-8B-DB-00-5E</epasd:Address>
843   <epasd:Connection>VM Network</epasd:Connection>
844   <epasd:Description>Virtual NIC</epasd:Description>
```

```

845     <epasd:ElementName>Ethernet Port 1</epasd:ElementName>
846     <epasd:InstanceID>3</epasd:InstanceID>
847     <epasd:NetworkPortProfileID>1</epasd:NetworkPortProfileID>
848     <epasd:NetworkPortProfileIDType>4</epasd:NetworkPortProfileIDType>
849     <epasd:VirtualQuantityUnits>1</epasd:VirtualQuantityUnits>
850 </EthernetPortItem>

```

851 The element type of the `StorageItem` element in a `VirtualHardwareSection` element is
 852 `CIM_StorageAllocationSettingData_Type` as defined in [http://schemas.dmtf.org/wbem/wscim/1/cim-](http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_StorageAllocationSettingData.xsd)
 853 [schema/2/CIM_StorageAllocationSettingData.xsd](http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_StorageAllocationSettingData.xsd).

854 The child elements represent the values of one or more properties exposed by the
 855 `CIM_StorageAllocationSettingData` class. They have the semantics of defined settings as defined
 856 in [DSP1047](#), any profiles derived from [DSP1047](#) for specific storage resource types, and this document.

857 EXAMPLE: The following example shows a description of a virtual storage:

```

858 <StorageItem>
859   <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
860   <sasd:Description>Virtual Disk</sasd:Description>
861   <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
862   <sasd:InstanceID>4</sasd:InstanceID>
863   <sasd:Reservation>100</sasd:Reservation>
864   <sasd:ResourceType>31</sasd:ResourceType>
865   <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
866 </StorageItem>

```

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

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

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

876 Devices such as disks, CD-ROMs, and networks need a backing from the deployment platform. The
 877 requirements on a backing are either specified using the `HostResource` or the `Connection` element.

878 For an Ethernet adapter, a logical network name is specified in the `Connection` element. Ethernet
 879 adapters that refer to the same logical network name within an OVF package shall be deployed on the
 880 same network.

881 The `HostResource` element is used to refer to resources included in the OVF descriptor as well as
 882 logical devices on the deployment platform. Values for `HostResource` elements referring to resources
 883 included in the OVF descriptor are formatted as URIs as specified in Table 3.

884 **Table 3 – HostResource Element**

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

885 If no backing is specified for a device that requires a backing, the deployment platform shall make an
 886 appropriate choice, for example, by prompting the user. More than one backing for a device shall not be
 887 specified.

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

890 **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. For example, "256MB memory".
InstanceID	A unique instance ID of the element within the section.
HostResource	Abstractly specifies how a device shall connect to a resource on the deployment 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	For an Ethernet adapter, this specifies the abstract network connection name for the virtual machine. All Ethernet adapters that specify the same abstract network connection name within an OVF package shall be deployed on the same network. The abstract network connection name shall be listed in the NetworkSection at the outermost envelope level.
Address	Device specific. For an Ethernet adapter, this specifies the MAC address.
AddressOnParent	For a device, this specifies its location on the controller.
AllocationUnits	Specifies the unit of allocation used. For example, "byte * 2^20".
VirtualQuantity	Specifies the quantity of resources presented. For example, "256".
Reservation	Specifies the minimum quantity of resources guaranteed to be available.
Limit	Specifies the maximum quantity of resources that are granted.
Weight	Specifies a relative priority for this allocation in relation to other allocations.

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

894 8.4 Ranges on Elements

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

898 A platform deploying an OVF package should start with the normal value and adjust the value within the
 899 range for ongoing performance tuning and validation.

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

- 902 • Each `Item`, `EthernetPortItem`, or `StorageItem` element has an optional `ovf:bound`
 903 attribute. This value may be specified as `min`, `max`, or `normal`. The value defaults to `normal`. If
 904 the attribute is not specified or is specified as `normal`, then the item shall be interpreted as
 905 being part of the regular virtual hardware or resource allocation description.

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

909 The semantics of range markers are as follows:

- 910 • InstanceID and ResourceType shall be specified, and the ResourceType shall match
 911 other Item elements with the same InstanceID.
- 912 • More than one `min` range marker nor more than one `max` range marker for a given RASD,
 913 EPASD, or SASD (identified with InstanceID) shall not be specified..
- 914 • An Item, EthernetPortItem, or StorageItem element with a range marker shall have
 915 a corresponding Item, EthernetPortItem, or StorageItem element without a range
 916 marker, that is, an Item, EthernetPortItem, and StorageItem element with no
 917 `ovf:bound` attribute or `ovf:bound` attribute with value `normal`. This corresponding item
 918 specifies the default value.
- 919 • For an Item, EthernetPortItem, and StorageItem element where only a `min` range
 920 marker is specified, the `max` value is unbounded upwards within the set of valid values for the
 921 property.
- 922 • For an Item, EthernetPortItem, and StorageItem where only a `max` range marker is
 923 specified, the `min` value is unbounded downwards within the set of valid values for the property.
- 924 • The default value shall be inside the range.
- 925 • Non-integer elements shall not be used in the range markers for RASD, EPASD, or SASD.

926 EXAMPLE: The following example shows the use of range markers:

```

927 <VirtualHardwareSection>
928   <Info>...</Info>
929   <Item>
930     <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
931     <rasd:ElementName>512 MB memory size</rasd:ElementName>
932     <rasd:InstanceID>0</rasd:InstanceID>
933     <rasd:ResourceType>4</rasd:ResourceType>
934     <rasd:VirtualQuantity>512</rasd:VirtualQuantity>
935   </Item>
936   <Item ovf:bound="min">
937     <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
938     <rasd:ElementName>384 MB minimum memory size</rasd:ElementName>
939     <rasd:InstanceID>0</rasd:InstanceID>
940     <rasd:Reservation>384</rasd:Reservation>
941     <rasd:ResourceType>4</rasd:ResourceType>
942   </Item>
943   <Item ovf:bound="max">
944     <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
945     <rasd:ElementName>1024 MB maximum memory size</rasd:ElementName>
946     <rasd:InstanceID>0</rasd:InstanceID>
947     <rasd:Reservation>1024</rasd:Reservation>
948     <rasd:ResourceType>4</rasd:ResourceType>
949   </Item>
950 </VirtualHardwareSection>
  
```

951

952 **9 Core Metadata Sections in version 2**953 Table 5 shows the core metadata sections that are defined in the `ovf` namespace.954 **Table 5 – Core Metadata Sections in version 2**

Section	Locations	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 installed guest operating system of 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> Specifies encryption scheme for encrypting parts of an OVF descriptor or files that it refers to.	Envelope	Zero or one

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

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

960 **9.1 DiskSection**

961 A `DiskSection` describes meta-information about virtual disks in the OVF package. Virtual disks and
962 their metadata are described outside the virtual hardware to facilitate sharing between virtual machines
963 within an OVF package. Virtual disks in `DiskSection` can be referenced by multiple virtual machines,
964 but seen from the guest software each virtual machine get individual private disks. Any level of sharing
965 done at runtime is deployment platform specific and not visible to the guest software. See clause 9.13 for
966 details on how to configure sharing of virtual disk at runtime with concurrent access.

967 **EXAMPLE:** The following example shows a description of virtual disks:

```
968 <DiskSection>  
969   <Info>Describes the set of virtual disks</Info>  
970   <Disk ovf:diskId="vmdisk1" ovf:fileRef="file1" ovf:capacity="8589934592"  
971     ovf:populatedSize="3549324972"  
972     ovf:format=  
973       "http://www.vmware.com/interfaces/specifications/vmdk.html#sparse">  
974   </Disk>  
975   <Disk ovf:diskId="vmdisk2" ovf:capacity="536870912"  
976   </Disk>  
977   <Disk ovf:diskId="vmdisk3" ovf:capacity="${disk.size}"  
978     ovf:capacityAllocationUnits="byte * 2^30"  
979   </Disk>  
980 </DiskSection>
```

981 `DiskSection` is a valid section at the outermost envelope level only.

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

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

989 The `ovf:fileRef` attribute denotes the virtual disk content by identifying an existing `File` element in
990 the `References` element, the `File` element is identified by matching its `ovf:id` attribute value with the
991 `ovf:fileRef` attribute value. Omitting the `ovf:fileRef` attribute shall indicate an empty disk. In this
992 case, the disk shall be created and the entire disk content zeroed at installation time. The guest software
993 will typically format empty disks in some file system format.

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

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

998 For empty disks, rather than specifying a fixed virtual disk capacity, the capacity for an empty disk may be
999 given using an OVF property, for example `ovf:capacity="${disk.size}"`. The OVF property shall
1000 resolve to an `xs:long` integer value. See 9.5 for a description of OVF properties. The
1001 `ovf:capacityAllocationUnits` attribute is useful when using OVF properties because a user may
1002 be prompted and can then enter disk sizing information in ,for example, gigabytes.

1003 For non-empty disks, the actual used size of the disk may optionally be specified using the
1004 `ovf:populatedSize` attribute. The unit of this attribute shall be bytes. The `ovf:populatedSize`
1005 attribute may be an estimate of used disk size but shall not be larger than `ovf:capacity`.

1006 In `VirtualHardwareSection`, virtual disk devices may have a `rasd:HostResource` element
 1007 referring to a `Disk` element in `DiskSection`; see 8.3. The virtual disk capacity shall be defined by the
 1008 `ovf:capacity` attribute on the `Disk` element. If a `rasd:VirtualQuantity` element is specified along
 1009 with the `rasd:HostResource` element, the virtual quantity value shall not be considered and may have
 1010 any value.

1011 OVF allows a disk image to be represented as a set of modified blocks in comparison to a parent image.
 1012 The use of parent disks can often significantly reduce the size of an OVF package if it contains multiple
 1013 disks with similar content, such as a common base operating system. Actual sharing of disk blocks at
 1014 runtime is optional and deployment platform specific and shall not be visible to the guest software.

1015 For the `Disk` element, a parent disk may optionally be specified using the `ovf:parentRef` attribute,
 1016 which shall contain a valid `ovf:diskId` reference to a different `Disk` element. If a disk block does not
 1017 exist locally, lookup for that disk block then occurs in the parent disk. In `DiskSection`, parent `Disk`
 1018 elements shall occur before child `Disk` elements that refer to them. Similarly, in `References` element,
 1019 the `File` elements referred from these `Disk` elements shall respect the same ordering. The ordering
 1020 restriction ensures that in an OVA archive, parent disks always occur before child disks, making it
 1021 possible for a tool to consume the archive in a streaming mode, see also clause 5.3.

1022 9.2 NetworkSection

1023 The `NetworkSection` element shall list all logical networks used in the OVF package.

```

1024 <NetworkSection>
1025   <Info>List of logical networks used in the package</Info>
1026   <Network ovf:name="VM Network">
1027     <Description>The network that the service will be available on</Description>
1028     <NetworkPortProfile>
1029       <Item>
1030         <epasd:AllocationUnits>GigaBits per Second</epasd:AllocationUnits>
1031         <epasd:ElementName>Network Port Profile 1</epasd:ElementName>
1032         <epasd:InstanceID>1</epasd:InstanceID>
1033         <epasd:NetworkPortProfileID>1</epasd:NetworkPortProfileID>
1034         <epasd:NetworkPortProfileIDType>4</epasd:NetworkPortProfileIDType>
1035         <epasd:Reservation>1</epasd:Reservation>
1036       </Item>
1037     </NetworkPortProfile>
1038   </Network>
1039 </NetworkSection>
  
```

1040 `NetworkSection` is a valid element at the outermost envelope level. A `Network` element is a child
 1041 element of `NetworkSection`. Each `Network` element in the `NetworkSection` shall be given a unique
 1042 name using the `ovf:name` attribute. The name shall be unique within an `ovf` envelope.

1043 All networks referred to from `Connection` elements in all `VirtualHardwareSection` elements shall
 1044 be defined in the `NetworkSection`.

1045 Starting with version 2.0 of this specification, each logical network may contain a set of networking
 1046 attributes that should be applied when mapping the logical network at deployment time to a physical or
 1047 virtual network. Networking attributes are specified by embedding or referencing zero or more instances
 1048 of network port profile as specified by `NetworkPortProfile` or `NetworkPortProfileURI` child
 1049 element of the `Network` element.

1050 The `NetworkPortProfile` child element of the `Network` element defines the contents of a network
 1051 port profile. The `NetworkPortProfileURI` child element of the `Network` element defines the
 1052 reference to a network port profile.

1053 Examples of using the DSP8049 and EPASD are in ANNEX D.

1054 9.3 ResourceAllocationSection

1055 The ResourceAllocationSection element describes all resource allocation requirements of a
 1056 VirtualSystemCollection entity. These resource allocations shall be performed when deploying the
 1057 OVF package.

```

1058 <ResourceAllocationSection>
1059   <Info>Defines reservations for CPU and memory for the collection of VMs</Info>
1060   <Item>
1061     <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1062     <rasd:ElementName>300 MB reservation</rasd:ElementName>
1063     <rasd:InstanceID>0</rasd:InstanceID>
1064     <rasd:Reservation>300</rasd:Reservation>
1065     <rasd:ResourceType>4</rasd:ResourceType>
1066   </Item>
1067   <Item ovf:configuration="..." ovf:bound="...">
1068     <rasd:AllocationUnits>hertz * 10^6</rasd:AllocationUnits>
1069     <rasd:ElementName>500 MHz reservation</rasd:ElementName>
1070     <rasd:InstanceID>0</rasd:InstanceID>
1071     <rasd:Reservation>500</rasd:Reservation>
1072     <rasd:ResourceType>3</rasd:ResourceType>
1073   </Item>
1074   <EthernetPortItem>
1075     <epasd:Address>00-16-8B-DB-00-5E</epasd:Address>
1076     <epasd:Connection>VM Network</epasd:Connection>
1077     <epasd:Description>Virtual NIC</epasd:Description>
1078     <epasd:ElementName>Ethernet Port 1</epasd:ElementName>
1079     <epasd:InstanceID>3</epasd:InstanceID>
1080     <epasd:NetworkPortProfileID>1</epasd:NetworkPortProfileID>
1081     <epasd:NetworkPortProfileIDType>4</epasd:NetworkPortProfileIDType>
1082     <epasd:VirtualQuantityUnits>1</epasd:VirtualQuantityUnits>
1083   </EthernetPortItem>
1084   <StorageItem>
1085     <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
1086     <sasd:Description>Virtual Disk</sasd:Description>
1087     <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
1088     <sasd:InstanceID>4</sasd:InstanceID>
1089     <sasd:Reservation>100</sasd:Reservation>
1090     <sasd:ResourceType>31</sasd:ResourceType>
1091     <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
1092   </StorageItem>
1093 </ResourceAllocationSection>

```

1094 ResourceAllocationSection is a valid element for a VirtualSystemCollection entity.

1095 The optional ovf:configuration attribute contains a list of configuration names. See 9.8 on
 1096 deployment options for semantics of this attribute.

1097 The optional ovf:bound attribute contains a value of min, max, or normal. See 8.4 for semantics of this
 1098 attribute.

1099 9.4 AnnotationSection

1100 The AnnotationSection element is a user-defined annotation on an entity. Such annotations may be
 1101 displayed when deploying the OVF package.

```

1102 <AnnotationSection>
1103   <Info>An annotation on this service. It can be ignored</Info>
1104   <Annotation>Contact customer support if you have any problems</Annotation>
1105 </AnnotationSection >

```

1106 AnnotationSection is a valid element for a VirtualSystem and a VirtualSystemCollection
1107 entity.

1108 See clause 10 for details on how to localize the Annotation element.

1109 9.5 ProductSection

1110 The ProductSection element specifies product-information for an appliance, such as product name,
1111 version, and vendor.

```
1112 <ProductSection ovf:class="com.mycrm.myservice" ovf:instance="1">
1113   <Info>Describes product information for the service</Info>
1114   <Product>MyCRM Enterprise</Product>
1115   <Vendor>MyCRM Corporation</Vendor>
1116   <Version>4.5</Version>
1117   <FullVersion>4.5-b4523</FullVersion>
1118   <ProductUrl>http://www.mycrm.com/enterprise</ProductUrl>
1119   <VendorUrl>http://www.mycrm.com</VendorUrl>
1120   <Icon ovf:height="32" ovf:width="32" ovf:mimeType="image/png" ovf:fileRef="icon">
1121   <Category>Email properties</Category>
1122   <Property ovf:key="admin.email" ovf:type="string" ovf:userConfigurable="true">
1123     <Label>Admin email</Label>
1124     <Description>Email address of administrator</Description>
1125   </Property>
1126   <Category>Admin properties</Category>
1127   <Property ovf:key="app_log" ovf:type="string" ovf:value="low"
1128   ovf:userConfigurable="true">
1129     <Description>Loglevel for the service</Description>
1130   </Property>
1131   <Property ovf:key="app_isSecondary" ovf:value="false" ovf:type="boolean">
1132     <Description>Cluster setup for application server</Description>
1133   </Property>
1134   <Property ovf:key="app_ip" ovf:type="string" ovf:value="{appserver-vm}">
1135     <Description>IP address of the application server VM</Description>
1136   </Property>
1137 </ProductSection>
```

1138 The optional Product element specifies the name of the product, while the optional Vendor element
1139 specifies the name of the product vendor. The optional Version element specifies the product version in
1140 short form, while the optional FullVersion element describes the product version in long form. The
1141 optional ProductUrl element specifies a URL which shall resolve to a human readable description of
1142 the product, while the optional VendorUrl specifies a URL which shall resolve to a human readable
1143 description of the vendor.

1144 The optional AppUrl element specifies a URL resolving to the deployed product instance. The optional
1145 Icon element specifies display icons for the product.

1146 The Property elements specify application-level customization parameters and are particularly relevant
1147 to appliances that need to be customized during deployment with specific settings such as network
1148 identity, the IP addresses of DNS servers, gateways, and others.

1149 The ProductSection is a valid section for a VirtualSystem and a VirtualSystemCollection entity.

1150 The Property elements may be grouped by using Category elements. The set of Property elements
1151 grouped by a Category element is the sequence of Property elements following the Category
1152 element, until but not including an element that is not a Property element. For OVF packages
1153 containing a large number of Property elements, this may provide a simpler installation experience.
1154 Similarly, each Property element may have a short label defined by its Label child element in addition

- 1155 to a description defined by its `Description` child element. See clause 10 for details on how to localize
1156 the `Category` element and the `Description` and `Label` child elements of the `Property` element.
- 1157 Each `Property` element in a `ProductSection` shall be given an identifier that is unique within the
1158 `ProductSection` using the `ovf:key` attribute.
- 1159 Each `Property` element in a `ProductSection` shall be given a type using the `ovf:type` attribute and
1160 optionally type qualifiers using the `ovf:qualifiers` attribute. Valid types are listed in Table 6, and valid
1161 qualifiers are listed in Table 7.
- 1162 The optional attribute `ovf:value` is used to provide a default value for a property. One or more optional
1163 `Value` elements may be used to define alternative default values for different configurations, as defined
1164 in 9.8.
- 1165 The optional attribute `ovf:userConfigurable` determines whether the property value is configurable
1166 during the installation phase. If `ovf:userConfigurable` is `FALSE` or omitted, the `ovf:value` attribute
1167 specifies the value to be used for that customization parameter during installation. If
1168 `ovf:userConfigurable` is `TRUE`, the `ovf:value` attribute specifies a default value for that
1169 customization parameter, which may be changed during installation.
- 1170 A simple OVF implementation such as a command-line installer typically uses default values for
1171 properties and does not prompt even though `ovf:userConfigurable` is set to `TRUE`. To force
1172 prompting at startup time, omitting the `ovf:value` attribute is sufficient for integer types, because the
1173 empty string is not a valid integer value. For string types, prompting may be forced by adding a qualifier
1174 requiring a non-empty string, see Table 7.
- 1175 The optional Boolean attribute `ovf:password` indicates that the property value may contain sensitive
1176 information. The default value is `FALSE`. OVF implementations prompting for property values are advised
1177 to obscure these values when `ovf:password` is set to `TRUE`. This is similar to HTML text input of type
1178 `password`. Note that this mechanism affords limited security protection only. Although sensitive values
1179 are masked from casual observers, default values in the OVF descriptor and assigned values in the OVF
1180 environment are still passed in clear text.
- 1181 Zero or more `ProductSections` may be specified within a `VirtualSystem` or
1182 `VirtualSystemCollection`. Typically, a `ProductSection` corresponds to a particular software
1183 product that is installed. Each product section at the same entity level shall have a unique `ovf:class`
1184 and `ovf:instance` attribute pair. For the common case where only a single `ProductSection` is used,
1185 the `ovf:class` and `ovf:instance` attributes are optional and default to the empty string. The
1186 `ovf:class` property should be used to uniquely identify the software product using the reverse domain
1187 name convention. Examples of values are `com.vmware.tools` and `org.apache.tomcat`. If multiple
1188 instances of the same product are installed, the `ovf:instance` attribute shall be used to identify the
1189 different instances.
- 1190 Property elements are exposed to the guest software through the OVF environment, as described in
1191 clause 11. The value of the `ovfenv:key` attribute of a `Property` element exposed in the OVF
1192 environment shall be constructed from the value of the `ovf:key` attribute of the corresponding
1193 `Property` element defined in a `ProductSection` entity of an OVF descriptor as follows:
- 1194

```
key-value-env = [class-value "."] key-value-prod ["." instance-value]
```
- 1195 The syntax definition above use ABNF with the exceptions listed in ANNEX A, where:
- 1196 • `class-value` is the value of the `ovf:class` attribute of the `Property` element defined in the
1197 `ProductSection` entity. The production `[class-value "."]` shall be present if and only if
1198 `class-value` is not the empty string.

- 1199 • `key-value-prod` is the value of the `ovf:key` attribute of the `Property` element defined in the
- 1200 `ProductSection` entity.
- 1201 • `instance-value` is the value of the `ovf:instance` attribute of the `Property` element defined in
- 1202 the `ProductSection` entity. The production `["." instance-value]` shall be present if and only
- 1203 if `instance-value` is not the empty string.

1204 EXAMPLE: The following OVF environment example shows how properties can be propagated to the guest
 1205 software:

```

1206 <Property ovf:key="com.vmware.tools.logLevel" ovf:value="none"/>
1207 <Property ovf:key="org.apache.tomcat.logLevel.1" ovf:value="debug"/>
1208 <Property ovf:key="org.apache.tomcat.logLevel.2" ovf:value="normal"/>
    
```

1209 The consumer of an OVF package should prompt for properties where `ovf:userConfigurable` is
 1210 TRUE. These properties may be defined in multiple `ProductSections` as well as in sub-entities in the
 1211 OVF package.

1212 If a `ProductSection` exists, then the first `ProductSection` entity defined in the top-level `Content`
 1213 element of a package shall define summary information that describes the entire package. After
 1214 installation, a consumer of the OVF package could choose to make this information available as an
 1215 instance of the `CIM_Product` class.

1216 `Property` elements specified on a `VirtualSystemCollection` are also seen by its immediate
 1217 children (see clause 11). Children may refer to the properties of a parent `VirtualSystemCollection`
 1218 using macros on the form `#{name}` as value for `ovf:value` attributes.

1219 Table 6 lists the valid types for properties. These are a subset of CIM intrinsic types defined in [DSP0004](#),
 1220 which also define the value space and format for each intrinsic type. Each `Property` element shall
 1221 specify a type using the `ovf:type` attribute.

1222 **Table 6 – Property Types**

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

1223 Table 7 lists the supported CIM type qualifiers as defined in [DSP0004](#). Each `Property` element may
 1224 optionally specify type qualifiers using the `ovf:qualifiers` attribute with multiple qualifiers separated
 1225 by commas; see production `qualifierList` in ANNEX A “MOF Syntax Grammar Description” in
 1226 [DSP0004](#).

1227

Table 7 – Property Qualifiers

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

1228 9.6 EulaSection

1229 A `EulaSection` contains the legal terms for using its parent `Content` element. This license shall be
 1230 shown and accepted during deployment of an OVF package. Multiple `EulaSections` may be present in
 1231 an OVF. If unattended installations are allowed, all embedded license sections are implicitly accepted.

```

1232 <EulaSection>
1233   <Info>Licensing agreement</Info>
1234   <License>
1235   Lorem ipsum dolor sit amet, ligula suspendisse nulla pretium, rhoncus tempor placerat
1236   fermentum, enim integer ad vestibulum volutpat. Nisl rhoncus turpis est, vel elit,
1237   congue wisi enim nunc ultricies sit, magna tincidunt. Maecenas aliquam maecenas ligula
1238   nostra, accumsan taciti. Sociis mauris in integer, a dolor netus non dui aliquet,
1239   sagittis felis sodales, dolor sociis mauris, vel eu libero cras. Interdum at. Eget
1240   habitasse elementum est, ipsum purus pede porttitor class, ut adipiscing, aliquet sed
1241   auctor, imperdiet arcu per diam dapibus libero duis. Enim eros in vel, volutpat nec
1242   pellentesque leo, scelerisque.
1243   </License>
1244 </EulaSection>
  
```

1245 The `EulaSection` is a valid section for a `VirtualSystem` and a `VirtualSystemCollection` entity.

1246 See clause 10 for details on how to localize the `License` element.

1247 See also clause 10 for description of storing EULA license contents in an external file without any XML
 1248 header or footer. This allows inclusion of standard license or copyright text files in unaltered form.

1249 9.7 StartupSection

1250 The `StartupSection` specifies how a virtual machine collection is powered on and off.

```

1251 <StartupSection>
1252   <Item ovf:id="vm1" ovf:order="0" ovf:startDelay="30" ovf:stopDelay="0"
1253   ovf:startAction="powerOn" ovf:waitingForGuest="true"
1254 ovf:stopAction="powerOff"/>
1255   <Item ovf:id="teamA" ovf:order="0"/>
1256   <Item ovf:id="vm2" ovf:order="1" ovf:startDelay="0" ovf:stopDelay="20"
1257   ovf:startAction="powerOn" ovf:stopAction="guestShutdown"/>
1258 </StartupSection>
  
```

1259 Each `Content` element that is a direct child of a `VirtualSystemCollection` may have a
 1260 corresponding `Item` element in the `StartupSection` entity of the `VirtualSystemCollection` entity.
 1261 Note that `Item` elements may correspond to both `VirtualSystem` and `VirtualSystemCollection`

1262 entities. When a start or stop action is performed on a `VirtualSystemCollection` entity, the
 1263 respective actions on the `Item` elements of its `StartupSection` entity are invoked in the specified
 1264 order. Whenever an `Item` element corresponds to a (nested) `VirtualSystemCollection` entity, the
 1265 actions on the `Item` elements of its `StartupSection` entity shall be invoked before the action on the
 1266 `Item` element corresponding to that `VirtualSystemCollection` entity is invoked (i.e., depth-first
 1267 traversal).

1268 The following required attributes on `Item` are supported for a `VirtualSystem` and
 1269 `VirtualSystemCollection`:

- 1270 • `ovf:id` shall match the value of the `ovf:id` attribute of a `Content` element which is a direct
 1271 child of this `VirtualSystemCollection`. That `Content` element describes the virtual
 1272 machine or virtual machine collection to which the actions defined in the `Item` element apply.
- 1273 • `ovf:order` specifies the startup order using non-negative integer values. If the `ovf:order`
 1274 = "0" then the order is not specified. If the `ovf:order` is non-zero then the of execution of the
 1275 start action shall be the numerical ascending order of the values. The `Items` with same order
 1276 identifier may be started concurrently.

1277 The order of execution of the stop action should be the numerical descending order of the
 1278 values. In implementation specific scenarios the order of execution of the stop action may be
 1279 non-descending.

1280 The following optional attributes on `Item` are supported for a `VirtualSystem`.

- 1281 • `ovf:startDelay` specifies a delay in seconds to wait until proceeding to the next order in the
 1282 start sequence. The default value is 0.
- 1283 • `ovf:waitingForGuest` enables the platform to resume the startup sequence after the guest
 1284 software has reported it is ready. The interpretation of this is deployment platform specific. The
 1285 default value is `FALSE`.
- 1286 • `ovf:startAction` specifies the start action to use. Valid values are `powerOn` and `none`. The
 1287 default value is `powerOn`.
- 1288 • `ovf:stopDelay` specifies a delay in seconds to wait until proceeding to the previous order in
 1289 the stop sequence. The default value is 0.
- 1290 • `ovf:stopAction` specifies the stop action to use. Valid values are `powerOff`,
 1291 `guestShutdown`, and `none`. The interpretation of `guestShutdown` is deployment platform
 1292 specific. The default value is `powerOff`.

1293 If the `StartupSection` is not specified then an `ovf:order="0"` is implied.

1294 9.8 DeploymentOptionSection

1295 The `DeploymentOptionSection` specifies a discrete set of intended resource configurations. The
 1296 author of an OVF package can include sizing metadata for different configurations. A consumer of the
 1297 OVF shall select a configuration, for example, by prompting the user. The selected configuration shall be
 1298 available in the OVF environment file, enabling the guest software to adapt to the selected configuration.
 1299 See clause 11.

1300 The `DeploymentOptionSection` specifies an ID, label, and description for each configuration.

```
1301 <DeploymentOptionSection>
1302   <Configuration ovf:id="minimal">
1303     <Label>Minimal</Label>
1304     <Description>Some description</Description>
1305   </Configuration>
1306   <Configuration ovf:id="normal" ovf:default="true">
1307     <Label>Typical</Label>
1308     <Description>Some description</Description>
```

```

1309     </Configuration>
1310     <!-- Additional configurations -->
1311 </DeploymentOptionSection>

```

1312 The `DeploymentOptionSection` has the following semantics:

- 1313 • If present, the `DeploymentOptionSection` is valid only at the envelope level, and only one
- 1314 section shall be specified in an OVF descriptor.
- 1315 • The discrete set of configurations is described with `Configuration` elements, which shall
- 1316 have identifiers specified by the `ovf:id` attribute that are unique in the package.
- 1317 • A default `Configuration` element may be specified with the optional `ovf:default` attribute.
- 1318 If no default is specified, the first element in the list is the default. Specifying more than one
- 1319 element as the default is invalid.
- 1320 • The `Label` and `Description` elements are localizable using the `ovf:msgid` attribute. See
- 1321 clause 10 for more details on internationalization support.

1322 Configurations may be used to control resources for virtual hardware and for virtual machine collections.

1323 `Item`, `EthernetPortItem`, and `StorageItem` elements in `VirtualHardwareSection` elements

1324 describe resources for `VirtualSystem` entities, while `Item`, `EthernetPortItem`, and `StorageItem`

1325 elements in `ResourceAllocationSection` elements describe resources for virtual machine

1326 collections. For these two `Item`, `EthernetPortItem`, or `StorageItem` types, the following

1327 additional semantics are defined:

- 1328 • Each `Item`, `EthernetPortItem`, and `StorageItem` has an optional
- 1329 `ovf:configuration` attribute, containing a list of configurations separated by a single space
- 1330 character. If not specified, the item shall be selected for any configuration. If specified, the item
- 1331 shall be selected only if the chosen configuration ID is in the list. A configuration attribute shall
- 1332 not contain an ID that is not specified in the `DeploymentOptionSection`.
- 1333 • Within a single `VirtualHardwareSection` or `ResourceAllocationSection`, multiple
- 1334 `Item`, `EthernetPortItem`, and `StorageItem` elements are allowed to refer to the same
- 1335 `InstanceID`. A single combined `Item`, `EthernetPortItem`, or `StorageItem` for the
- 1336 given `InstanceID` shall be constructed by picking up the child elements of each `Item`,
- 1337 `EthernetPortItem`, or `StorageItem` element, with child elements of a former `Item`,
- 1338 `EthernetPortItem`, or `StorageItem` element in the OVF descriptor not being picked up
- 1339 if there is a like-named child element in a latter `Item`, `EthernetPortItem`, or
- 1340 `StorageItem` element. Any attributes specified on child elements of `Item`,
- 1341 `EthernetPortItem`, or `StorageItem` elements that are not picked up that way, are not
- 1342 part of the combined `Item`, `EthernetPortItem`, or `StorageItem` element.
- 1343 • All `Item`, `EthernetPortItem`, `StorageItem` elements shall specify `ResourceType`, and
- 1344 `Item`, `EthernetPortItem`, and `StorageItem` elements with the same `InstanceID` shall
- 1345 agree on `ResourceType`.

1346 EXAMPLE 1: The following example shows a `VirtualHardwareSection`:

```

1347 <VirtualHardwareSection>
1348   <Info>...</Info>
1349   <Item>
1350     <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1351     <rasd:ElementName>512 MB memory size and 256 MB
1352 reservation</rasd:ElementName>
1353     <rasd:InstanceID>0</rasd:InstanceID>
1354     <rasd:Reservation>256</rasd:Reservation>
1355     <rasd:ResourceType>4</rasd:ResourceType>
1356     <rasd:VirtualQuantity>512</rasd:VirtualQuantity>
1357   </Item>
1358   ...
1359   <Item ovf:configuration="big">
1360     <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>

```

```

1361     <rasd:ElementName>1024 MB memory size and 512 MB
1362 reservation</rasd:ElementName>
1363     <rasd:InstanceID>0</rasd:InstanceID>
1364     <rasd:Reservation>512</rasd:Reservation>
1365     <rasd:ResourceType>4</rasd:ResourceType>
1366     <rasd:VirtualQuantity>1024</rasd:VirtualQuantity>
1367   </Item>
1368 </VirtualHardwareSection>

```

1369 Note that the attributes `ovf:configuration` and `ovf:bound` on `Item` may be used in combination to
 1370 provide very flexible configuration options.

1371 Configurations can further be used to control default values for properties and whether properties are
 1372 user configurable. For `Property` elements inside a `ProductSection`, the following additional semantic
 1373 is defined:

- 1374 • It is possible to specify alternative default property values for different configurations in a
 1375 `DeploymentOptionSection`. In addition to a `Label` and `Description` element, each
 1376 `Property` element may optionally contain `Value` elements. The `Value` element shall have
 1377 an `ovf:value` attribute specifying the alternative default and an `ovf:configuration`
 1378 attribute specifying the configuration in which this new default value should be used. Multiple
 1379 `Value` elements shall not refer to the same configuration.
- 1380 • Starting with version 2.0 of this specification, a `Property` element may optionally have an
 1381 `ovf:configuration` attribute specifying the configuration in which this property should be
 1382 user configurable. The value of `ovf:userConfigurable` is implicitly set to `FALSE` for all
 1383 other configurations, in which case the default value of the property may not be changed
 1384 during installation.

1385 **EXAMPLE 2:** The following shows an example `ProductSection`:

```

1386 <ProductSection>
1387   <Property ovf:key="app.adminEmail" ovf:type="string" ovf:userConfigurable="true"
1388     ovf:configuration="standard">
1389     <Label>Admin email</Label>
1390     <Description>Email address of service administrator</Description>
1391   </Property>
1392   <Property ovf:key="app.log" ovf:type="string" ovf:value="low"
1393     ovf:userConfigurable="true">
1394     <Label>Loglevel</Label>
1395     <Description>Loglevel for the service</Description>
1396     <Value ovf:value="none" ovf:configuration="minimal">
1397   </Property>
1398 </ProductSection>

```

1399 In the example above, the `app.adminEmail` property is only user configurable in the `standard`
 1400 configuration, while the default value for the `app.log` property is changed from `low` to `none` in the
 1401 `minimal` configuration.

1402 9.9 OperatingSystemSection

1403 An `OperatingSystemSection` specifies the operating system installed on a virtual machine.

```

1404 <OperatingSystemSection ovf:id="76">
1405   <Info>Specifies the operating system installed</Info>
1406   <Description>Microsoft Windows Server 2008</Description>
1407 </OperatingSystemSection>

```

1408 The values for `ovf:id` should be taken from the ValueMap of the `CIM_OperatingSystem.OsType`
1409 property. The description should be taken from the corresponding Values of the
1410 `CIM_OperatingSystem.OsType` property.

1411 The `OperatingSystemSection` is a valid section for a `VirtualSystem` entity only.

1412 9.10 InstallSection

1413 The `InstallSection`, if specified, indicates that the virtual machine needs to be booted once in order
1414 to install and/or configure the guest software. The guest software is expected to access the OVF
1415 environment during that boot, and to shut down after having completed the installation and/or
1416 configuration of the software, powering off the guest.

1417 If the `InstallSection` is not specified, this indicates that the virtual machine does not need to be
1418 powered on to complete installation of guest software.

```
1419 <InstallSection ovf:initialBootStopDelay="300">  
1420   <Info>Specifies that the virtual machine needs to be booted once after having  
1421   created the guest software in order to install and/or configure the software  
1422   </Info>  
1423 </InstallSection>
```

1424 `InstallSection` is a valid section for a `VirtualSystem` entity only.

1425 The optional `ovf:initialBootStopDelay` attribute specifies a delay in seconds to wait for the virtual
1426 machine to power off. If not set, the implementation shall wait for the virtual machine to power off by itself.
1427 If the delay expires and the virtual machine has not powered off, the consumer of the OVF package shall
1428 indicate a failure.

1429 Note that the guest software in the virtual machine can do multiple reboots before powering off.

1430 Several VMs in a virtual machine collection may have an `InstallSection` defined, in which case the
1431 above step is done for each VM, potentially concurrently.

1432 9.11 EnvironmentFilesSection

1433 Clause 11 describes how the OVF environment file is used to deliver runtime customization parameters to
1434 the guest operating system. In version 1 of this specification, the OVF environment file is the only file
1435 delivered to the guest operating system outside of the virtual disk structure. In order to provide additional
1436 deployment time customizations, the `EnvironmentFilesSection` enables the OVF package authors
1437 to specify additional files in the OVF package, outside of the virtual disks, that also is provided to the
1438 guest operating system at runtime via a transport.

1439 This enables increased flexibility in image customization outside of virtual disk capture, allowing OVF
1440 package authors to customize solutions by combining existing virtual disks without modifying them.

1441 For each additional file provided to the guest, the `EnvironmentFilesSection` shall contain a `File`
1442 element with required attributes `ovf:fileRef` and `ovf:path`. The `ovf:fileRef` attribute shall denote
1443 the actual content by identifying an existing `File` element in the `References` element, the `File`
1444 element is identified by matching its `ovf:id` attribute value with the `ovf:fileRef` attribute value. The
1445 `ovf:path` attribute denotes the relative location on the transport where this file will be placed, using the
1446 syntax of relative-path references in [RFC3986](#).

1447 The referenced `File` element in the `References` element identify the content using one of the URL
1448 schemes "file", "http", or "https". For the "file" scheme, the content is static and included in
1449 the OVF package. For the "http" and "https" schemes, the content shall be downloaded prior to the
1450 initial boot of the virtual system.

1451 The `iso` transport shall support this mechanism, see clause 11.2 for details. For this transport, the root
 1452 location relative to `ovf:path` values shall be directory `ovffiles` contained in the root directory of the
 1453 ISO image. The guest software can access the information using standard guest operating system tools.

1454 Other custom transport may support this mechanism. Custom transports will need to specify how to
 1455 access multiple data sources from a root location.

1456 EXAMPLE:

```

1457 <Envelope>
1458   <References>
1459     ...
1460     <File ovf:id="config" ovf:href="config.xml" ovf:size="4332"/>
1461     <File ovf:id="resources" ovf:href="http://mywebsite/resources/resources.zip"/>
1462   </References>
1463   ...
1464   <VirtualSystem ovf:id="...">
1465     ...
1466     <ovf:EnvironmentFilesSection ovf:required="false" ovf:transport="iso">
1467       <Info>Config files to be included in OVF environment</Info>
1468       <ovf:File ovf:fileRef="config" ovf:path="setup/cfg.xml"/>
1469       <ovf:File ovf:fileRef="resources" ovf:path="setup/resources.zip"/>
1470     </ovf:EnvironmentFilesSection>
1471     ...
1472   </VirtualSystem>
1473   ...
1474 </Envelope>

```

1475 In the example above, the file `config.xml` in the OVF package will be copied to the OVF environment
 1476 ISO image and be accessible to the guest software in location `/ovffiles/setup/cfg.xml`, while the
 1477 file `resources.zip` will be accessible in location `/ovffiles/setup/resources.zip`.

1478 9.12 BootDeviceSection

1479 Individual virtual machines will generally use the default device boot order provided by the virtualization
 1480 platform's virtual BIOS. The `BootDeviceSection` allows the OVF package author to specify particular
 1481 boot configurations and boot order settings. This enables booting from non-default devices such as a NIC
 1482 using PXE, a USB device or a secondary disk. Moreover there could be multiple boot configurations with
 1483 different boot orders. For example, a virtual disk may be need to be patched before it is bootable and a
 1484 patch ISO image could be included in the OVF package.

1485 The Common Information Model (CIM) defines artifacts to deal with boot order use cases prevalent in the
 1486 industry for BIOSes found in desktops and servers. The boot configuration is defined by the class
 1487 `CIM_BootConfigSetting` which in turn contains one or more `CIM_BootSourceSetting` classes as
 1488 defined in the WS-CIM schema. Each class representing the boot source in turn has either the specific
 1489 device or a "device type" such as disk or CD/DVD as a boot source.

1490 In the context of this specification, the `InstanceID` field of `CIM_BootSourceSetting` is used for
 1491 identifying a specific device as the boot source. The `InstanceID` field of the device as specified in the
 1492 `Item` description of the device in the `VirtualHardwareSection` is used to specify the device as a
 1493 boot source. In case the source is desired to be a device type, the `StructuredBootString` field is
 1494 used to denote the type of device with values defined by the CIM boot control profile. When a boot source
 1495 is a device type, the deployment platform should try all the devices of the specified type.

1496 In the example below, the Pre-Install configuration specifies the boot source as a specific device
 1497 (network), while the Post-Install configuration specifies a device type (hard disk).

1498 EXAMPLE:

```

1499 <Envelope>
1500 ...
1501 <VirtualSystem ovf:id="...">
1502 ...
1503 <ovf:BootDeviceSection>
1504 <Info>Boot device order specification</Info>
1505 <bootc:CIM_BootConfigSetting>
1506 <bootc:Caption>Pre-Install</bootc:Caption>
1507 <bootc:Description>Boot Sequence for fixup of disk</bootc:Description>
1508 <boots:CIM_BootSourceSetting>
1509 <boots:Caption>Fix-up DVD on the network</boots:Caption>
1510 <boots:InstanceID>3</boots:InstanceID> <!-- Network device-->
1511 </boots:CIM_BootSourceSetting>
1512 <boots:CIM_BootSourceSetting>
1513 <boots:Caption>Boot virtual disk</boots:Caption>
1514 <boots:StructuredBootString>CIM:Hard-Disk</boots:StructuredBootString>
1515 </boots:CIM_BootSourceSetting>
1516 </bootc:CIM_BootConfigSetting>
1517 </ovf:BootDeviceSection>
1518 ...
1519 </VirtualSystem>
1520 </Envelope>

```

1521 9.13 SharedDiskSection

1522 The existing `DiskSection` in clause 9.1 describes virtual disks in the OVF package. Virtual disks in the
 1523 `DiskSection` can be referenced by multiple virtual machines, but seen from the guest software each
 1524 virtual machine gets individual private disks. Any level of sharing done at runtime is deployment platform
 1525 specific and not visible to the guest software.

1526 Certain applications such as clustered databases rely on multiple virtual machines sharing the same
 1527 virtual disk at runtime. `SharedDiskSection` allows the OVF package author to specify `Disk` elements
 1528 shared by more than one `VirtualSystem` at runtime, these could be virtual disks backing by an external
 1529 `File` reference, or empty virtual disks without backing. It is recommended that the guest software use
 1530 cluster-aware file system technology to be able to handle concurrent access.

1531 EXAMPLE:

```

1532 <ovf:SharedDiskSection>
1533 <Info>Describes the set of virtual disks shared between VMs</Info>
1534 <ovf:SharedDisk ovf:diskId="datadisk" ovf:fileRef="data"
1535 <ovf:capacity="8589934592" ovf:populatedSize="3549324972"
1536 <ovf:format=
1537 "http://www.vmware.com/interfaces/specifications/vmdk.html#sparse"/>
1538 <ovf:SharedDisk ovf:diskId="transientdisk" ovf:capacity="536870912"/>
1539 </ovf:SharedDiskSection>

```

1540 `SharedDiskSection` is a valid section at the outermost envelope level only.

1541 Each virtual disk is represented by a `SharedDisk` element that shall be given an identifier using the
 1542 `ovf:diskId` attribute; the identifier shall be unique within the combined content of `DiskSection` and
 1543 `SharedDiskSection`. The `SharedDisk` element has the same structure as the `Disk` element in
 1544 `DiskSection`, with the addition of an optional boolean attribute `ovf:readOnly` stating whether shared
 1545 disk access is read-write or read-only.

1546 Shared virtual disks are referenced from virtual hardware using the `HostResource` element as described
 1547 in clause 8.3.

1548 It is optional for the virtualization platform to support `SharedDiskSection`. The platform should give an
 1549 appropriate error message based on the value of the `ovf:required` attribute on the
 1550 `SharedDiskSection` element.

1551 9.14 ScaleOutSection

1552 The number of `VirtualSystems` and `VirtualSystemCollections` contained in an OVF package is generally
 1553 fixed and determined by the structure inside the `Envelope` element. The `ScaleOutSection` allows a
 1554 `VirtualSystemCollection` to contain a set of children that are homogeneous with respect to a prototypical
 1555 `VirtualSystem` or `VirtualSystemCollection`. The `ScaleOutSection` shall cause the deployment platform
 1556 to replicate the prototype a number of times, thus allowing the number of instantiated virtual machines to
 1557 be configured dynamically at deployment time.

1558 EXAMPLE:

```
1559 <VirtualSystemCollection ovf:id="web-tier">
1560   ...
1561   <ovf:ScaleOutSection ovf:id="web-server">
1562     <Info>Web tier</Info>
1563     <ovf:Description>Number of web server instances in web tier</ovf:Description>
1564     <ovf:InstanceCount ovf:default="4" ovf:minimum="2" ovf:maximum="8"/>
1565   </ovf:ScaleOutSection>
1566   ...
1567   <VirtualSystem ovf:id="web-server">
1568     <Info>Prototype web server</Info>
1569     ...
1570   </VirtualSystem>
1571 </VirtualSystemCollection>
```

1572 In the example above, the deployment platform creates a web tier containing between two and eight web
 1573 server virtual machine instances, with a default instance count of four. The deployment platform makes
 1574 an appropriate choice (e.g., by prompting the user). Assuming three replicas were created, the OVF
 1575 environment available to the guest software in the first replica has the following content structure:

1576 EXAMPLE:

```
1577 <Environment ... ovfenv:id="web-server-1">
1578   ...
1579   <Entity ovfenv:id="web-server-2">
1580     ...
1581   </Entity>
1582   <Entity ovfenv:id="web-server-3">
1583     ...
1584   </Entity>
1585 </Environment>
```

1586 This mechanism enables dynamic scaling of virtual machine instances at deployment time. Scaling at
 1587 runtime is not within the scope of this specification.

1588 The `ScaleOutSection` is a valid section inside `VirtualSystemCollection` only.

1589 The `ovf:id` attribute on `ScaleOutSection` identifies the `VirtualSystem` or `VirtualSystemCollection`
 1590 prototype to be replicated.

1591 For the `InstanceCount` element, the `ovf:minimum` and `ovf:maximum` attribute values shall be non-
 1592 negative integers and `ovf:minimum` shall be less than or equal to the value of `ovf:maximum`. The
 1593 `ovf:minimum` value may be zero in which case the `VirtualSystemCollection` may contain zero instances
 1594 of the prototype. If the `ovf:minimum` attribute is not present, it shall be assumed to have a value of one.
 1595 If the `ovf:maximum` attribute is not present, it shall be assumed to have a value of unbounded. The
 1596 `ovf:default` attribute is required and shall contain a value within the range defined by `ovf:minimum`
 1597 and `ovf:maximum`.

1598 Each replicated instance shall be assigned a unique `ovf:id` value within the `VirtualSystemCollection`.
 1599 The unique `ovf:id` value shall be constructed from the prototype `ovf:id` value with a sequence
 1600 number appended as follows:

```
1601 replica-ovf-id = prototype-ovf-id "-" decimal-number
1602 decimal-number = decimal-digit | (decimal-digit decimal-number)
1603 decimal-digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
```

1604 The syntax definitions above use ABNF with the exceptions listed in ANNEX A. The first replica shall
 1605 have sequence number one and following sequence numbers shall be incremented by one for each
 1606 replica. Note that after deployment, no `VirtualSystem` will have the prototype `ovf:id` value itself.

1607 If the prototype being replicated has a starting order in the `StartupSection`, all replicas shall share this
 1608 value. It is not possible to specify a particular starting sequence among replicas.

1609 Property values for Property elements in the prototype are prompted for once per replica created. If the
 1610 OVF package author requires a property value to be shared among instances, that Property may be
 1611 declared at the containing `VirtualSystemCollection` level and referenced by replicas as described in
 1612 clause 9.5.

1613 Configurations from the `DeploymentOptionSection` may be used to control values for `InstanceCount`. The
 1614 `InstanceCount` element may have an `ovf:configuration` attribute specifying the configuration in
 1615 which this element should be used. Multiple elements shall not refer to the same configuration, and a
 1616 configuration attribute shall not contain an `ovf:id` value that is not specified in the
 1617 `DeploymentOptionSection`.

1618 EXAMPLE:

```
1619 <VirtualSystemCollection ovf:id="web-tier">
1620   ...
1621   <DeploymentOptionSection>
1622     <Info>Deployment size options</Info>
1623     <Configuration ovf:id="minimal">
1624       <Label>Minimal</Label>
1625       <Description>Minimal deployment scenario</Description>
1626     </Configuration>
1627     <Configuration ovf:id="common" ovf:default="true">
1628       <Label>Typical</Label>
1629       <Description>Common deployment scenario</Description>
1630     </Configuration>
1631     ...
1632   </DeploymentOptionSection>
1633   ...
1634   <ovf:ScaleOutSection ovf:id="web-server">
1635     <Info>Web tier</Info>
1636     <ovf:Description>Number of web server instances in web tier</ovf:Description>
1637     <ovf:InstanceCount ovf:default="4"/>
1638     <ovf:InstanceCount ovf:default="1" ovf:configuration="minimal"/>
1639   </ovf:ScaleOutSection>
1640   ...
1641 </VirtualSystemCollection>
```

1642 In the example above, the default replica count is four, unless the minimal deployment scenario is
 1643 chosen, in which case the default is one.

1644 9.15 PlacementGroupSection and PlacementSection

1645 Certain types of applications require the ability to specify that two or more `VirtualSystems` should be
 1646 deployed closely together since they rely on very fast communication or a common hardware dependency
 1647 such as a reliable communication link. Other types of applications require the ability to specify that two or

1648 more VirtualSystems should be deployed apart due to high-availability or disaster recovery
1649 considerations.

1650 PlacementGroupSection allow an OVF package author to define a placement policy for a group of
1651 VirtualSystems, while PlacementSection allow the author to annotate elements with membership of a
1652 particular placement policy group.

1653 Zero or more PlacementGroupSections may be declared at the Envelope level, while
1654 PlacementSection may be declared at the VirtualSystem or VirtualSystemCollection level. Declaring a
1655 VirtualSystemCollection member of a placement policy group applies transitively to all child VirtualSystem
1656 and child Virtual System Collections elements. The ovf:id attribute on PlacementGroupSection is
1657 used to identify the particular placement policy; the attribute value shall be unique within the OVF
1658 package. Placement policy group membership is specified using the ovf:group attribute on
1659 PlacementSection; the attribute value shall match the value of an ovf:id attribute on a
1660 PlacementGroupSection.

1661 This version of the specification defines the placement policies "affinity" and "availability",
1662 specified with the required ovf:policy attribute on PlacementGroupSection.

1663 Placement policy "affinity" describe that VirtualSystems should be placed as closely together as
1664 possible. The deployment platform should attempt to keep these virtual machines located as adjacently
1665 as possible, typically on the same physical host or with fast network connectivity between hosts.

1666 Placement policy "availability" describe that VirtualSystems should be placed separately. The
1667 deployment platform should attempt to keep these virtual machines located apart, typically on the
1668 different physical hosts.

1669 EXAMPLE:

```
1670 <Envelope ...>
1671   ...
1672   <ovf:PlacementGroupSection ovf:id="web" ovf:policy="availability">
1673     <Info>Placement policy for group of VMs</Info>
1674     <ovf:Description>Placement policy for web tier</ovf:Description>
1675   </ovf:PlacementGroupSection>
1676   ...
1677   <VirtualSystemCollection ovf:id="web-tier">
1678     ...
1679     <ovf:ScaleOutSection ovf:id="web-node">
1680       <Info>Web tier</Info>
1681       ...
1682     </ovf:ScaleOutSection>
1683     ...
1684     <VirtualSystem ovf:id="web-node">
1685       <Info>Web server</Info>
1686       ...
1687       <ovf:PlacementSection ovf:group="web">
1688         <Info>Placement policy group reference</Info>
1689       </ovf:PlacementSection>
1690       ...
1691     </VirtualSystem>
1692   </VirtualSystemCollection>
1693 </Envelope>
```

1694 In the example above, all virtual machines in the compute tier should be placed separately for high
1695 availability. This example also use the ScaleOutSection defined in clause 9.14, in which case each
1696 replica get the policy assigned.

1697 **9.16 Encryption Section**

1698 For licensing and other reasons it is desirable to have an encryption scheme enabling free exchange of
 1699 OVF appliances while ensuring that only the intended parties can use them. The XML Encryption Syntax
 1700 and Processing standard is utilized to encrypt either the files in the reference section or any parts of the
 1701 XML markup of an OVF document.

1702 The various aspects of OVF encryption are as shown below:

- 1703 1. block encryption
 1704 The OVF document author shall utilize block encryption algorithms as specified in the XML
 1705 encryption 1.1 documents (ref) for this purpose.
- 1706 2. key derivation
 1707 The OVF author may use the appropriate key for this purpose. If the key is derived using a
 1708 passphrase then the author shall use one of the key derivations specified in the XML Encryption
 1709 1.1 standard.
- 1710 3. key transport.
 1711 If the encryption key is embedded in the OVF document, the specified key transport mechanisms
 1712 shall be used.

1713 This specification defines a new section called the EncryptionSection as a focal point for the encryption
 1714 functionality. This new section provides a single location for placing the encryption algorithm related
 1715 markup and the corresponding reference list to point to the OVF content that has been encrypted.

1716 Note that depending on which parts of the OVF markup has been encrypted, an OVF descriptor may not
 1717 validate against the OVF schemas until decrypted.

1718 Below is an example of an OVF encryption section with encryption methods utilized in the OVF
 1719 document, and the corresponding reference list pointing to the items that have been encrypted.

1720 **EXAMPLE:**

```

1721 <ovf:EncryptionSection>
1722 <!-- This section contains two different methods of encryption and the corresponding
1723 backpointers to the data that is encrypted ->
1724 <!-- Method#1: Pass phrase based Key derivation ->
1725 <!-- The following derived key block defines PBKDF2 and the corresponding back
1726 pointers to the encrypted data elements -->
1727 <!-- Use a salt value "ovfpassword" and iteration count of 4096 --->
1728 <xenc11:DerivedKey>
1729 <xenc11:KeyDerivationMethod
1730 Algorithm="http://www.rsasecurity.com/rsalabs/pkcs/schemas/pkcs-5#pbkdf2"/>
1731 <pkcs-5:PBKDF2-params>
1732 <Salt>
1733 <Specified>ovfpassword</Specified>
1734 </Salt>
1735 <IterationCount>4096</IterationCount>
1736 <KeyLength>16</KeyLength>
1737 <PRF Algorithm="http://www.w3.org/2001/04/xmldsig-more#hmac-sha256"/>
1738 </pkcs-5:PBKDF2-params>
1739 ...
1740 <!-- The ReferenceList element below contains references to the file Ref-109.vhd via
1741 the URI syntax which is specified by XML Encryption.
1742 --->
1743 <xenc:ReferenceList>
1744 <xenc:DataReference URI="#first.vhd" />
1745 <xenc:DataReference URI=... />
1746 <xenc:DataReference URI=... />
1747 </xenc:ReferenceList>
1748 </xenc11:DerivedKey>
1749 <!-- Method#2: The following example illustrates use of a symmetric key
1750 transported using the public key within a certificate ->
  
```

```

1751 <xenc:EncryptedKey>
1752   <xenc:EncryptionMethod   Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-
1753   1_5"/>
1754     <ds:KeyInfo xmlns:ds='http://www.w3.org/2000/09/xmldsig#'
1755       <ds:X509Data>
1756         <ds:X509Certificate> ... </ds:X509Certificate>
1757       </ds:X509Data>
1758     </ds:KeyInfo>
1759     <xenc:CipherData>
1760       <xenc:CipherValue> ... </xenc:CipherValue>
1761     </xenc:CipherData>
1762 <!-- The ReferenceList element below contains reference #second-xml-fragment" to the
1763 XML fragment that has been encrypted using the above method --->
1764   <xenc:ReferenceList>
1765     <xenc:DataReference URI='#second-xml-fragment' />
1766     <xenc:DataReference URI='...' />
1767     <xenc:DataReference URI='...' />
1768   </xenc:ReferenceList>
1769 </xenc:EncryptedKey>
1770 </ovf:EncryptionSection>

```

1771 Below is an example of the encrypted file which is referenced in the EncryptionSection above using
 1772 URI='Ref-109.vhd' syntax.

1773 **EXAMPLE:**

```

1774 <ovf:References>
1775 <ovf:File ovf:id="Xen:9cb10691-4012-4aeb-970c-3d47a906bfff/0b13bdba-3761-8622-22fc-
1776 2e252ed9ce14" ovf:href="Ref-109.vhd">
1777 <!-- the encrypted file referenced by the package is enclosed by an EncryptedData with
1778 a CipherReference to the actual encrypted file. The EncryptionSection in this example
1779 has a back pointer to it under the PBKDF2 algorithm via Id="first.vhd". This tells the
1780 decrypter how to decrypt the file -->
1781 <xenc:EncryptedData Id="first.vhd" Type='http://www.w3.org/2001/04/xmlenc#Element' >
1782   <xenc:EncryptionMethod
1783     Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc" />
1784     <xenc:CipherData>
1785       <xenc:CipherReference URI='Ref-109.vhd' />
1786     </xenc:CipherData>
1787   </xenc:EncryptedData>
1788 </ovf:File>
1789 </ovf:References>

```

1790 Below is an example of the encrypted OVF markup which is referenced in the EncryptionSection above
 1791 using URI='#second-xml-fragment' syntax.

1792 **EXAMPLE:**

```

1793 <!-- the EncryptedData element below encompasses encrypted xml from the original
1794 document. It is provided with the Id "first-xml-fragment" which allows it to be
1795 referenced from the EncryptionSection. -->
1796 <xenc:EncryptedData Type=http://www.w3.org/2001/04/xmlenc#Element Id="second-xml-
1797 fragment">
1798 <!-- Each EncryptedData specifies its own encryption method. -->
1799   <xenc:EncryptionMethod Algorithm=http://www.w3.org/2001/04/xmlenc#aes128-cbc/>
1800   <xenc:CipherData>
1801     <!-- Encrypted content --->
1802     <xenc:CipherValue>DEADBEEF</xenc:CipherValue>
1803   </xenc:CipherData>
1804 </xenc:EncryptedData>

```

1805 10 Internationalization

1806 The following elements support localizable messages using the optional `ovf:msgid` attribute:

- 1807 • Info element on Content
- 1808 • Name element on Content
- 1809 • Info element on Section
- 1810 • Annotation element on AnnotationSection
- 1811 • License element on EulaSection
- 1812 • Description element on NetworkSection
- 1813 • Description element on OperatingSystemSection
- 1814 • Description, Product, Vendor, Label, and Category elements on ProductSection
- 1815 • Description and Label elements on Property
- 1816 • Description and Label elements on DeploymentOptionSection
- 1817 • ElementName, Caption and Description subelements on the System element in
- 1818 VirtualHardwareSection
- 1819 • ElementName, Caption and Description subelements on Item elements in
- 1820 VirtualHardwareSection
- 1821 • ElementName, Caption and Description subelements on Item elements in
- 1822 ResourceAllocationSection

1823 The `ovf:msgid` attribute contains an identifier that refers to a message that may have different values in
1824 different locales.

1825 EXAMPLE 1:

```
1826 <Info ovf:msgid="info.text">Default info.text value if no locale is set or no locale
1827 match</Info>
1828 <License ovf:msgid="license.tomcat-6_0"/> <!-- No default message -->
```

1829 The `xml:lang` attribute on the `Envelope` element shall specify the default locale for messages in the
1830 descriptor. The attribute is optional with a default value of "en-US".

1831 10.1 Internal Resource Bundles

1832 Message resource bundles can be internal or external to the OVF descriptor. Internal resource bundles
1833 are represented as `Strings` elements at the end of the `Envelope` element.

1834 EXAMPLE 2:

```
1835 <ovf:Envelope xml:lang="en-US">
1836   ...
1837   ... sections and content here ...
1838   ...
1839   <Info msgid="info.os">Operating System</Info>
1840   ...
1841   <Strings xml:lang="da-DA">
1842     <Msg ovf:msgid="info.os">Operativsystem</Msg>
1843     ...
1844   </Strings>
1845   <Strings xml:lang="de-DE">
1846     <Msg ovf:msgid="info.os">Betriebssystem</Msg>
1847     ...
1848   </Strings>
1849 </ovf:Envelope>
```

1850 10.2 External Resource Bundles

1851 External resource bundles shall be listed first in the `References` section and referred to from `Strings`
 1852 elements. An external message bundle follows the same schema as the embedded one. Exactly one
 1853 `Strings` element shall be present in an external message bundle, and that `Strings` element may not
 1854 have an `ovf:fileRef` attribute specified.

1855 EXAMPLE 3:

```
1856 <ovf:Envelope xml:lang="en-US">
1857   <References>
1858     ...
1859     <File ovf:id="it-it-resources" ovf:href="resources/it-it-bundle.msg"/>
1860   </References>
1861   ... sections and content here ...
1862   ...
1863   <Strings xml:lang="it-IT" ovf:fileRef="it-it-resources"/>
1864   ...
1865 </ovf:Envelope>
```

1866 EXAMPLE 4: Example content of external `resources/it-it-bundle.msg` file, which is referenced in previous example:

```
1867 <Strings
1868   xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1"
1869   xmlns="http://schemas.dmtf.org/ovf/envelope/1"
1870   xml:lang="it-IT">
1871   <Msg ovf:msgid="info.os">Sistema operativo</Msg>
1872   ...
1873 </Strings>
```

1874 The embedded and external `Strings` elements may be interleaved, but they shall be placed at the end
 1875 of the `Envelope` element. If multiple occurrences of a `msgid` attribute with a given locale occur, a latter
 1876 value overwrites a former.

1877 10.3 Message Content in External File

1878 Starting with version 2.0 of this specification, the content of all localizable messages may be stored in an
 1879 external file using the optional `ovf:fileRef` attribute on the `Msg` element. For the `License` element on
 1880 `EulaSection` in particular, this allows inclusion of a standard license text file in unaltered form without
 1881 any XML header or footer.

1882 The `ovf:fileRef` attribute denotes the message content by identifying an existing `File` element in the
 1883 `References` element, the `File` element is identified by matching its `ovf:id` attribute value with the
 1884 `ovf:fileRef` attribute value. The content of an external file referenced using `ovf:fileRef` shall be
 1885 interpreted as plain text in UTF-8 Unicode.

1886 If the referenced file is not found, the embedded content of the `Msg` element shall be used.

1887 The optional `ovf:fileRef` attribute may appear on `Msg` elements in both internal and external `Strings`
 1888 resource bundles.

1889 EXAMPLE 5:

```
1890 <Envelope xml:lang="en-US">
1891   <References>
1892     <File ovf:id="license-en-US" ovf:href="license-en-US.txt"/>
1893     <File ovf:id="license-de-DE" ovf:href="license-de-DE.txt"/>
1894   </References>
1895   ...
1896   <VirtualSystem ovf:id="...">
1897     <EulaSection>
1898       <Info>Licensing agreement</Info>
1899       <License ovf:msgid="license">Unused</License>
```

```

1900     </EulaSection>
1901     ...
1902 </VirtualSystem>
1903     ...
1904     <Strings xml:lang="en-US">
1905         <Msg ovf:msgid="license" ovf:fileRef="license-en-US">Invalid license</Msg>
1906     </Strings>
1907     <Strings xml:lang="de-DE">
1908         <Msg ovf:msgid="license" ovf:fileRef="license-de-DE">Ihre Lizenz ist nicht
1909 gültig</Msg>
1910     </Strings>
1911 </Envelope>

```

1912 In the example above, the default license agreement is stored in plain text file `license-en-US.txt`,
 1913 while the license agreement for the `de-DE` locale is stored in file `license-de-DE.txt`.

1914 Note that the above mechanism works for all localizable elements and not just `License`.

1915 11 OVF Environment

1916 The OVF environment defines how the guest software and the deployment platform interact. This
 1917 environment allows the guest software to access information about the deployment platform, such as the
 1918 user-specified values for the properties defined in the OVF descriptor.

1919 The environment specification is split into a *protocol* part and a *transport* part. The *protocol* part defines
 1920 the format and semantics of an XML document that can be made accessible to the guest software. The
 1921 *transport* part defines how the information is communicated between the deployment platform and the
 1922 guest software.

1923 The `dsp8027_1.1.0.xsd` XML schema definition file for the OVF environment contains the elements
 1924 and attributes.

1925 11.1 Environment Document

1926 The environment document is an extensible XML document that is provided to the guest software about
 1927 the environment in which it is being executed. The way that the document is obtained depends on the
 1928 transport type.

1929 **EXAMPLE:** An example of the structure of the OVF environment document follows:

```

1930 <?xml version="1.0" encoding="UTF-8"?>
1931 <Environment xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1932     xmlns:ovfenv="http://schemas.dmtf.org/ovf/environment/1"
1933     xmlns="http://schemas.dmtf.org/ovf/environment/1"
1934     ovfenv:id="identification of VM from OVF descriptor">
1935     <!-- Information about virtualization platform -->
1936     <PlatformSection>
1937         <Kind>Type of virtualization platform</Kind>
1938         <Version>Version of virtualization platform</Version>
1939         <Vendor>Vendor of virtualization platform</Vendor>
1940         <Locale>Language and country code</Locale>
1941         <TimeZone>Current timezone offset in minutes from UTC</TimeZone>
1942     </PlatformSection>
1943     <!-- Properties defined for this virtual machine -->
1944     <PropertySection>
1945         <Property ovfenv:key="key" ovfenv:value="value">
1946             <!-- More properties -->
1947         </Property>
1948     </PropertySection>
1949     <Entity ovfenv:id="id of sibling virtual system or virtual system collection">
1950         <PropertySection>
1951             <!-- Properties from sibling -->

```



```

1951     </PropertySection>
1952   </Entity>
1953 </Environment>
    
```

1954 The value of the `ovfenv:id` attribute of the `Environment` element shall match the value of the `ovf:id`
 1955 attribute of the `VirtualSystem` entity describing this virtual machine.

1956 The `PlatformSection` element contains optional information provided by the deployment platform.
 1957 Elements `Kind`, `Version`, and `Vendor` describe deployment platform vendor details; these elements are
 1958 experimental. Elements `Locale` and `TimeZone` describe the current locale and time zone; these
 1959 elements are experimental.

1960 The `PropertySection` element contains `Property` elements with key/value pairs corresponding to all
 1961 properties specified in the OVF descriptor for the current virtual machine, as well as properties specified
 1962 for the immediate parent `VirtualSystemCollection`, if one exists. The environment presents
 1963 properties as a simple list to make it easy for applications to parse. Furthermore, the single list format
 1964 supports the override semantics where a property on a `VirtualSystem` may override one defined on a
 1965 parent `VirtualSystemCollection`. The overridden property shall not be in the list. Overriding may
 1966 occur if a property in the current virtual machine and a property in the parent
 1967 `VirtualSystemCollection` has identical `ovf:key`, `ovf:class`, and `ovf:instance` attribute
 1968 values; see 9.5. In this case, the value of an overridden parent property may be obtained by adding a
 1969 differently named child property referencing the parent property with a macro; see 9.5.

1970 An `Entity` element shall exist for each sibling `VirtualSystem` and `VirtualSystemCollection`, if
 1971 any are present. The value of the `ovfenv:id` attribute of the `Entity` element shall match the value of
 1972 the `ovf:id` attribute of the sibling entity. The `Entity` elements contain the property key/value pairs in
 1973 the sibling's OVF environment documents, so the content of an `Entity` element for a particular sibling
 1974 shall contain the exact `PropertySection` seen by that sibling. This information can be used, for
 1975 example, to make configuration information such as IP addresses available to `VirtualSystems` being
 1976 part of a multi-tiered application.

1977 Table 8 shows the core sections that are defined.

1978 **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

1979 The environment document is extensible by providing new section types. A consumer of the document
 1980 should ignore unknown section types and elements.

1981 11.2 Transport

1982 The environment document information can be communicated in a number of ways to the guest software.
 1983 These ways are called transport types. The transport types are specified in the OVF descriptor by the
 1984 `ovf:transport` attribute of `VirtualHardwareSection`. Several transport types may be specified,
 1985 separated by a single space character, in which case an implementation is free to use any of them. The
 1986 transport types define methods by which the environment document is communicated from the
 1987 deployment platform to the guest software.

1988 To enable interoperability, this specification defines an "iso" transport type which all implementations
 1989 that support CD-ROM devices are required to support. The `iso` transport communicates the environment
 1990 document by making a dynamically generated ISO image available to the guest software. To support the

- 1991 `iso` transport type, prior to booting a virtual machine, an implementation shall make an ISO read-only
1992 disk image available as backing for a disconnected CD-ROM. If the `iso` transport is selected for a
1993 `VirtualHardwareSection`, at least one disconnected CD-ROM device shall be present in this section.
- 1994 The generated ISO image shall comply with the ISO 9660 specification with support for Joliet extensions.
- 1995 The ISO image shall contain the OVF environment for this particular virtual machine, and the environment
1996 shall be present in an XML file named `ovf-env.xml` that is contained in the root directory of the ISO
1997 image. The guest software can now access the information using standard guest operating system tools.
- 1998 If the virtual machine prior to booting had more than one disconnected CD-ROM, the guest software may
1999 have to scan connected CD-ROM devices in order to locate the ISO image containing the `ovf-env.xml`
2000 file.
- 2001 The ISO image containing the OVF environment shall be made available to the guest software on every
2002 boot of the virtual machine.
- 2003 Support for the "`iso`" transport type is not a requirement for virtual hardware architectures or guest
2004 operating systems which do not have CD-ROM device support.
- 2005 To be compliant with this specification, any transport format other than `iso` shall be given by a URI which
2006 identifies an unencumbered specification on how to use the transport. The specification need not be
2007 machine readable, but it shall be static and unique so that it may be used as a key by software reading an
2008 OVF descriptor to uniquely determine the format. The specification shall be sufficient for a skilled person
2009 to properly interpret the transport mechanism for implementing the protocols. The URIs should be
2010 resolvable.

ANNEX A (informative)

2011
2012
2013
2014

Symbols and Conventions

2015 XML examples use the XML namespace prefixes defined in Table 1. The XML examples use a style to
2016 not specify namespace prefixes on child elements. Note that XML rules define that child elements
2017 specified without namespace prefix are from the namespace of the parent element, and not from the
2018 default namespace of the XML document. Throughout the document, whitespace within XML element
2019 values is used for readability. In practice, a service can accept and strip leading and trailing whitespace
2020 within element values as if whitespace had not been used.

2021 Syntax definitions in this document use Augmented BNF (ABNF) as defined in IETF [RFC5234](#) with the
2022 following exceptions:

- 2023 • Rules separated by a bar (|) represent choices, instead of using a forward slash (/) as defined in
2024 ABNF.
- 2025 • Any characters must be processed case sensitively, instead of case-insensitively as defined in
2026 ABNF.
- 2027 • Whitespace (i.e., the space character U+0020 and the tab character U+0009) is allowed between
2028 syntactical elements, instead of assembling elements without whitespace as defined in ABNF.

2029

ANNEX B (normative)

OVF XSD

2030
2031
2032
2033

2034 Normative copies of the XML schemas for this specification may be retrieved by resolving the following
2035 URLs:

2036
2037
2038

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

2039 Any `xs:documentation` content in XML schemas for this specification is informative and provided only
2040 for convenience.

2041 Normative copies of the XML schemas for the WS-CIM mapping ([DSP0230](#)) of
2042 `CIM_ResourceAllocationSystemSettingsData`, `CIM_VirtualSystemSettingData`,
2043 `CIM_EthernetPortAllocationSettingData`, `CIM_StorageAllocationSettingData` and
2044 `CIM_OperatingSystem`, may be retrieved by resolving the following URLs:

2045
2046
2047
2048
2049
2050
2051

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

2052 This specification is based on the following CIM MOFs:

2053
2054
2055
2056
2057
2058

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

ANNEX C (informative)

OVF Mime Type Registration Template

2059
2060
2061
2062

2063 Registration Template

2064 To: ietf-types@iana.org

2065 Subject: Registration of media type Application/OVF

2066 Type name: Application

2067 Subtype name: OVF

2068 Required parameters: none

2069 Optional parameters: none

2070 Encoding considerations: binary

2071 Security considerations:

- 2072 • An OVF package contains active content that is expected to be launched in a virtual machine.
2073 The OVF standard, section 5.1 states: “An OVF package may be signed by signing the manifest
2074 file. The digest of the manifest file is stored in a certificate file with extension .cert file along with
2075 the base64-encoded X.509 certificate. The .cert file shall have the same base name as the .ovf
2076 file and be a sibling of the .ovf file. A consumer of the OVF package shall verify the signature and
2077 should validate the certificate.
- 2078 • An OVF package may contain passwords as part of the configuration information. The OVF
2079 standard, section 9.5 states: “The optional Boolean attribute ovf:password indicates that the
2080 property value may contain sensitive information. The default value is FALSE. OVF
2081 implementations prompting for property values are advised to obscure these values when
2082 ovf:password is set to TRUE. This is similar to HTML text input of type password. Note that this
2083 mechanism affords limited security protection only. Although sensitive values are masked from
2084 casual observers, default values in the OVF descriptor and assigned values in the OVF
2085 environment are still passed in clear text. “

2086 Interoperability considerations:

- 2087 • OVF has demonstrated interoperability via multiple, interoperating implementations in the market.

2088 Published specification:

- 2089 • DSP0243_2.0.0.pdf

2090 Applications that use this media type:

- 2091 • Implementations of the DMTF Standard: Cloud Infrastructure Management Interface (CIMI)
2092 (DSP0263_1.0.0.pdf)
- 2093 • Implementations of the SNIA Cloud Data Management Interface (CDMI) – OVF Extension

2094 Additional information:

- 2095 • Magic number(s): none
- 2096 • File extension(s): .ova
- 2097 • Macintosh file type code(s): none
- 2098 • Person & email address to contact for further information:

- 2099 • Intended usage: (One of COMMON, LIMITED USE or OBSOLETE.)
- 2100 • Restrictions on usage: (Any restrictions on where the media type can be used go here.)
- 2101 • Author:
- 2102 • Change controller:

ANNEX D (informative)

Network Port Profile Examples

D.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.

```

2114 <?xml version="1.0" encoding="UTF-8"?>
2115 <Envelope xsi:schemaLocation="http://schemas.dmtf.org/ovf/envelope/2
2116 file:///C:/dsp8023_2.0.0_wgv0.9.5.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2117 xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/2" xmlns="http://schemas.dmtf.org/ovf/envelope/2"
2118 xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
2119 xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
2120 xmlns:epasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2121 schema/2/CIM_EthernetPortAllocationSettingData"
2122 xmlns:sasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_StorageAllocationSettingData">
2123 <!-- References to all external files -->
2124 <References>
2125 <File ovf:id="file1" ovf:href="vmdisk1.vmdk" ovf:size="2000000000"/>
2126 </References>
2127 <!-- Describes meta-information for all virtual disks in the package -->
2128 <DiskSection>
2129 <Info>Describes the set of virtual disks</Info>
2130 <Disk ovf:diskId="vmdisk1" ovf:fileRef="file1" ovf:capacity="4294967296"
2131 ovf:format="http://www.examplecompany.com/interfaces/specifications/vmdk.html#sparse"/>
2132 </DiskSection>
2133 <!-- Describes all networks used in the package -->
2134 <NetworkSection>
2135 <Info>List of logical networks used in the package</Info>
2136 <Network ovf:name="VM Network">
2137 <Description>The network that the VMs connect to</Description>
2138 <NetworkPortProfile>
2139 <!-- Network port profile describing bandwidth reservation. Network port profile
2140 is identified by UUID. -->
2141 <Item>
2142 <epasd:AllocationUnits>bit / second * 10^9</epasd:AllocationUnits>
2143 <epasd:ElementName>Network Port Profile 1</epasd:ElementName>
2144 <epasd:InstanceID>1</epasd:InstanceID>
2145 <epasd:NetworkPortProfileID>aaaaaaaa-bbbb-cccc-dddd-
2146 eeeeeeeeeee</epasd:NetworkPortProfileID>
2147 <epasd:NetworkPortProfileIDType>3</epasd:NetworkPortProfileIDType>
2148 <epasd:Reservation>1</epasd:Reservation>
2149 </Item>
2150 </NetworkPortProfile>
2151 </Network>
2152 </NetworkSection>
2153 <VirtualSystem ovf:id="vm">
2154 <Info>Describes a virtual machine</Info>
2155 <Name>Virtual Appliance One</Name>
2156 <ProductSection>
2157 <Info>Describes product information for the appliance</Info>
2158 <Product>The Great Appliance</Product>
2159 <Vendor>Some Great Corporation</Vendor>
2160 <Version>13.00</Version>
2161 <FullVersion>13.00-b5</FullVersion>
2162 <ProductUrl>http://www.somegreatcorporation.com/greatappliance</ProductUrl>
2163 <VendorUrl>http://www.somegreatcorporation.com/</VendorUrl>
2164 <Property ovf:key="admin.email" ovf:type="string">

```

```

2165         <Description>Email address of administrator</Description>
2166     </Property>
2167     <Property ovf:key="app.ip" ovf:type="string" ovf:defaultValue="192.168.0.10">
2168         <Description>The IP address of this appliance</Description>
2169     </Property>
2170 </ProductSection>
2171 <AnnotationSection ovf:required="false">
2172     <Info>A random annotation on this service. It can be ignored</Info>
2173     <Annotation>Contact customer support if you have any problems</Annotation>
2174 </AnnotationSection>
2175 <EulaSection>
2176     <Info>License information for the appliance</Info>
2177     <License>Insert your favorite license here</License>
2178 </EulaSection>
2179 <VirtualHardwareSection>
2180     <Info>Memory = 4 GB, CPU = 1 GHz, Disk = 100 GB, 1 Ethernet nic</Info>
2181     <Item>
2182         <rasd:AllocationUnits>Hertz*10^9</rasd:AllocationUnits>
2183         <rasd:Description>Virtual CPU</rasd:Description>
2184         <rasd:ElementName>1 GHz virtual CPU</rasd:ElementName>
2185         <rasd:InstanceID>1</rasd:InstanceID>
2186         <rasd:Reservation>1</rasd:Reservation>
2187         <rasd:ResourceType>3</rasd:ResourceType>
2188         <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2189     </Item>
2190     <Item>
2191         <rasd:AllocationUnits>byte*2^30</rasd:AllocationUnits>
2192         <rasd:Description>Memory</rasd:Description>
2193         <rasd:ElementName>1 GByte of memory</rasd:ElementName>
2194         <rasd:InstanceID>2</rasd:InstanceID>
2195         <rasd:ResourceType>4</rasd:ResourceType>
2196         <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2197     </Item>
2198     <EthernetPortItem>
2199         <epasd:Address>00-16-8B-DB-00-5E</epasd:Address>
2200         <epasd:AllocationUnits>bit / second * 10^9 </epasd:AllocationUnits>
2201         <epasd:Connection>VM Network</epasd:Connection>
2202         <epasd:Description>Virtual NIC</epasd:Description>
2203         <epasd:ElementName>Ethernet Port</epasd:ElementName>
2204
2205         <epasd:InstanceID>3</epasd:InstanceID>
2206         <epasd:NetworkPortProfileID>aaaaaaa-bbbb-cccc-dddd-
2207 eeeeeeeee</epasd:NetworkPortProfileID>
2208         <epasd:NetworkPortProfileIDType>3</epasd:NetworkPortProfileIDType>
2209         <epasd:Reservation>1</epasd:Reservation>
2210         <epasd:ResourceType>10</epasd:ResourceType>
2211         <epasd:VirtualQuantityUnits>1</epasd:VirtualQuantityUnits>
2212     </EthernetPortItem>
2213     <StorageItem>
2214         <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
2215         <sasd:Description>Virtual Disk</sasd:Description>
2216         <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
2217         <sasd:InstanceID>4</sasd:InstanceID>
2218         <sasd:Reservation>100</sasd:Reservation>
2219         <sasd:ResourceType>31</sasd:ResourceType>
2220         <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
2221     </StorageItem>
2222 </VirtualHardwareSection>
2223 <OperatingSystemSection ovf:id="58" ovf:required="false">
2224     <Info>Guest Operating System</Info>
2225     <Description>OS</Description>
2226 </OperatingSystemSection>
2227 </VirtualSystem>
2228 </Envelope>

```


2229 D.2 Example 2 (OVF Descriptor for One Virtual System and One Network with a 2230 Locally Referenced Network Port Profile)

2231 The example below shows an OVF descriptor that describes a virtual system and a network it connects
2232 to. The virtual system description in this example uses a network port profile that is described in a local
2233 file that is contained in the same OVF package. The network described in the network section uses the
2234 same network port profile description. The network port profile described in this example is used to
2235 reserve 1 Gbps of bandwidth.

```

2236 <?xml version="1.0" encoding="UTF-8"?>
2237 <Envelope xsi:schemaLocation="http://schemas.dmtf.org/ovf/envelope/2
2238 file:///C:/dsp8023_2.0.0_wgv0.9.5.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2239 xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/2" xmlns="http://schemas.dmtf.org/ovf/envelope/2"
2240 xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
2241 xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
2242 xmlns:epasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2243 schema/2/CIM_EthernetPortAllocationSettingData"
2244 xmlns:sasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_StorageAllocationSettingData">
2245 <!-- References to all external files -->
2246 <References>
2247 <File ovf:id="file1" ovf:href="vmdisk1.vmdk" ovf:size="2000000000"/>
2248 <File ovf:id="networkportprofile1" ovf:href="NetworkPortProfile1.xml"/>
2249 </References>
2250 <!-- Describes meta-information for all virtual disks in the package -->
2251 <DiskSection>
2252 <Info>Describes the set of virtual disks</Info>
2253 <Disk ovf:diskId="vmdisk1" ovf:fileRef="file1" ovf:capacity="4294967296"
2254 ovf:format="http://www.examplecompany.com/interfaces/specifications/vmdk.html#sparse"/>
2255 </DiskSection>
2256 <!-- Describes all networks used in the package -->
2257 <NetworkSection>
2258 <Info>List of logical networks used in the package</Info>
2259 <Network ovf:name="VM Network">
2260 <Description>The network that VMs connect to</Description>
2261 <NetworkPortProfileURI>file:networkportprofile1</NetworkPortProfileURI>
2262 </Network>
2263 </NetworkSection>
2264 <VirtualSystem ovf:id="vm">
2265 <Info>Describes a virtual machine</Info>
2266 <Name>Virtual Appliance One</Name>
2267 <ProductSection>
2268 <Info>Describes product information for the appliance</Info>
2269 <Product>The Great Appliance</Product>
2270 <Vendor>Some Great Corporation</Vendor>
2271 <Version>13.00</Version>
2272 <FullVersion>13.00-b5</FullVersion>
2273 <ProductUrl>http://www.somegreatcorporation.com/greatappliance</ProductUrl>
2274 <VendorUrl>http://www.somegreatcorporation.com/</VendorUrl>
2275 <Property ovf:key="admin.email" ovf:type="string">
2276 <Description>Email address of administrator</Description>
2277 </Property>
2278 <Property ovf:key="app.ip" ovf:type="string" ovf:defaultValue="192.168.0.10">
2279 <Description>The IP address of this appliance</Description>
2280 </Property>
2281 </ProductSection>
2282 <AnnotationSection ovf:required="false">
2283 <Info>A random annotation on this service. It can be ignored</Info>
2284 <Annotation>Contact customer support if you have any problems</Annotation>
2285 </AnnotationSection>
2286 <EulaSection>
2287 <Info>License information for the appliance</Info>
2288 <License>Insert your favorite license here</License>
2289 </EulaSection>
2290 <VirtualHardwareSection>
2291 <Info>Memory = 4 GB, CPU = 1 GHz, Disk = 100 GB, 1 Ethernet nic</Info>
2292 <Item>
2293 <rasd:AllocationUnits>Hertz*10^9</rasd:AllocationUnits>
2294 <rasd:Description>Virtual CPU</rasd:Description>
2295 <rasd:ElementName>1 GHz virtual CPU</rasd:ElementName>

```

```

2296         <rasd:InstanceID>1</rasd:InstanceID>
2297         <rasd:Reservation>1</rasd:Reservation>
2298         <rasd:ResourceType>3</rasd:ResourceType>
2299         <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2300     </Item>
2301     <Item>
2302         <rasd:AllocationUnits>byte*2^30</rasd:AllocationUnits>
2303         <rasd:Description>Memory</rasd:Description>
2304         <rasd:ElementName>1 GByte of memory</rasd:ElementName>
2305         <rasd:InstanceID>2</rasd:InstanceID>
2306         <rasd:ResourceType>4</rasd:ResourceType>
2307         <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2308     </Item>
2309     <EthernetPortItem>
2310         <epasd:Address>00-16-8B-DB-00-5E</epasd:Address>
2311         <epasd:Connection>VM Network</epasd:Connection>
2312         <epasd:Description>Virtual NIC</epasd:Description>
2313         <epasd:ElementName>Ethernet Port</epasd:ElementName>
2314
2315         <epasd:InstanceID>3</epasd:InstanceID>
2316         <epasd:NetworkPortProfileID>file:networkportprofile1</epasd:NetworkPortProfileID>
2317         <epasd:NetworkPortProfileIDType>2</epasd:NetworkPortProfileIDType>
2318         <epasd:ResourceType>10</epasd:ResourceType>
2319         <epasd:VirtualQuantityUnits>1</epasd:VirtualQuantityUnits>
2320     </EthernetPortItem>
2321     <StorageItem>
2322         <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
2323         <sasd:Description>Virtual Disk</sasd:Description>
2324         <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
2325         <sasd:InstanceID>4</sasd:InstanceID>
2326         <sasd:Reservation>100</sasd:Reservation>
2327         <sasd:ResourceType>31</sasd:ResourceType>
2328         <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
2329     </StorageItem>
2330 </VirtualHardwareSection>
2331 <OperatingSystemSection ovf:id="58" ovf:required="false">
2332     <Info>Guest Operating System</Info>
2333     <Description>OS</Description>
2334 </OperatingSystemSection>
2335 </VirtualSystem>
2336 </Envelope>

```

2337 **D.3 Example 3 (OVF Descriptor for One Virtual System and One Network with a** 2338 **Network Port Profile referenced by a URI)**

2339 The example below shows an OVF descriptor that describes a virtual system and a network it connects
2340 to. The virtual system description in this example uses a network port profile that is described by a URI.
2341 The network described in the network section uses the same network port profile description. The
2342 network port profile described in this example is used to reserve 1 Gbps of bandwidth.

```

2343 <?xml version="1.0" encoding="UTF-8"?>
2344 <Envelope xsi:schemaLocation="http://schemas.dmtf.org/ovf/envelope/2
2345 file:///C:/dsp8023_2.0.0_wgv0.9.5.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2346 xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/2" xmlns="http://schemas.dmtf.org/ovf/envelope/2"
2347 xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
2348 xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
2349 xmlns:epasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2350 schema/2/CIM_EthernetPortAllocationSettingData"
2351 xmlns:sasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_StorageAllocationSettingData">
2352 <!-- References to all external files -->
2353 <References>
2354     <File ovf:id="file1" ovf:href="vmdisk1.vmdk" ovf:size="2000000000"/>
2355 </References>
2356 <!-- Describes meta-information for all virtual disks in the package -->
2357 <DiskSection>
2358     <Info>Describes the set of virtual disks</Info>
2359     <Disk ovf:diskId="vmdisk1" ovf:fileRef="file1" ovf:capacity="4294967296"
2360 ovf:format="http://www.examplecompany.com/interfaces/specifications/vmdk.html#sparse"/>
2361 </DiskSection>

```

```

2362 <!-- Describes all networks used in the package -->
2363 <NetworkSection>
2364   <Info>List of logical networks used in the package</Info>
2365   <Network ovf:name="VM Network">
2366     <Description>The network that the VMs connect to</Description>
2367
2368   <NetworkPortProfileURI>http://www.dmtf.org/networkportprofiles/networkportprofile1.xml</Netwo
2369 rkPortProfileURI>
2370   </Network>
2371 </NetworkSection>
2372 <VirtualSystem ovf:id="vm">
2373   <Info>Describes a virtual machine</Info>
2374   <Name>Virtual Appliance One</Name>
2375   <ProductSection>
2376     <Info>Describes product information for the appliance</Info>
2377     <Product>The Great Appliance</Product>
2378     <Vendor>Some Great Corporation</Vendor>
2379     <Version>13.00</Version>
2380     <FullVersion>13.00-b5</FullVersion>
2381     <ProductUrl>http://www.somegreatcorporation.com/greatappliance</ProductUrl>
2382     <VendorUrl>http://www.somegreatcorporation.com/</VendorUrl>
2383     <Property ovf:key="admin.email" ovf:type="string">
2384       <Description>Email address of administrator</Description>
2385     </Property>
2386     <Property ovf:key="app.ip" ovf:type="string" ovf:defaultValue="192.168.0.10">
2387       <Description>The IP address of this appliance</Description>
2388     </Property>
2389   </ProductSection>
2390   <AnnotationSection ovf:required="false">
2391     <Info>A random annotation on this service. It can be ignored</Info>
2392     <Annotation>Contact customer support if you have any problems</Annotation>
2393   </AnnotationSection>
2394   <EulaSection>
2395     <Info>License information for the appliance</Info>
2396     <License>Insert your favorite license here</License>
2397   </EulaSection>
2398   <VirtualHardwareSection>
2399     <Info>Memory = 4 GB, CPU = 1 GHz, Disk = 100 GB, 1 Ethernet nic</Info>
2400     <Item>
2401       <rasd:AllocationUnits>Hertz*10^9</rasd:AllocationUnits>
2402       <rasd:Description>Virtual CPU</rasd:Description>
2403       <rasd:ElementName>1 GHz virtual CPU</rasd:ElementName>
2404       <rasd:InstanceID>1</rasd:InstanceID>
2405       <rasd:Reservation>1</rasd:Reservation>
2406       <rasd:ResourceType>3</rasd:ResourceType>
2407       <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2408     </Item>
2409     <Item>
2410       <rasd:AllocationUnits>byte*2^30</rasd:AllocationUnits>
2411       <rasd:Description>Memory</rasd:Description>
2412       <rasd:ElementName>1 GByte of memory</rasd:ElementName>
2413       <rasd:InstanceID>2</rasd:InstanceID>
2414       <rasd:ResourceType>4</rasd:ResourceType>
2415       <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2416     </Item>
2417     <EthernetPortItem>
2418       <epasd:Address>00-16-8B-DB-00-5E</epasd:Address>
2419       <epasd:Connection>VM Network</epasd:Connection>
2420       <epasd:Description>Virtual NIC</epasd:Description>
2421       <epasd:ElementName>Ethernet Port</epasd:ElementName>
2422
2423       <epasd:InstanceID>3</epasd:InstanceID>
2424
2425       <epasd:NetworkPortProfileID>http://www.dmtf.org/networkportprofiles/networkportprofile1.xml</
2426 epasd:NetworkPortProfileID>
2427       <epasd:NetworkPortProfileIDType>2</epasd:NetworkPortProfileIDType>
2428       <epasd:ResourceType>10</epasd:ResourceType>
2429       <epasd:VirtualQuantityUnits>1</epasd:VirtualQuantityUnits>
2430     </EthernetPortItem>
2431     <StorageItem>
2432       <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>

```

```

2433         <sasd:Description>Virtual Disk</sasd:Description>
2434         <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
2435         <sasd:InstanceID>4</sasd:InstanceID>
2436         <sasd:Reservation>100</sasd:Reservation>
2437         <sasd:ResourceType>31</sasd:ResourceType>
2438         <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
2439     </StorageItem>
2440 </VirtualHardwareSection>
2441 <OperatingSystemSection ovf:id="58" ovf:required="false">
2442     <Info>Guest Operating System</Info>
2443     <Description>OS</Description>
2444 </OperatingSystemSection>
2445 </VirtualSystem>
2446 </Envelope>

```

2447 **D.4 Example 4 (OVF Descriptor for Two Virtual Systems and One Network with** 2448 **Two Network Port Profiles referenced by URIs)**

2449 The example below shows an OVF descriptor that describes two virtual systems and a network they
2450 connect to. Each virtual system description in this example uses a network port profile that is described
2451 by a URI. The network described in the network section uses the same two network port profiles. The two
2452 network port profiles described in this example are used to reserve 1 Gbps of bandwidth and describe
2453 general network traffic respectively. Annex D.5 and D.6 are examples of these network port profiles.

```

2454 <?xml version="1.0" encoding="UTF-8"?>
2455 <Envelope xsi:schemaLocation="http://schemas.dmtf.org/ovf/envelope/2
2456 file:///C:/dsp8023_2.0.0_wgv0.9.5.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2457 xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/2" xmlns="http://schemas.dmtf.org/ovf/envelope/2"
2458 xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
2459 xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
2460 xmlns:epasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2461 schema/2/CIM_EthernetPortAllocationSettingData"
2462 xmlns:sasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_StorageAllocationSettingData">
2463 <!-- References to all external files -->
2464 <References>
2465     <File ovf:id="file1" ovf:href="vmdisk1.vmdk" ovf:size="2000000000"/>
2466 </References>
2467 <!-- Describes meta-information for all virtual disks in the package -->
2468 <DiskSection>
2469     <Info>Describes the set of virtual disks</Info>
2470     <Disk ovf:diskId="vmdisk1" ovf:fileRef="file1" ovf:capacity="4294967296"
2471 ovf:format="http://www.examplecompany.com/interfaces/specifications/vmdk.html#sparse"/>
2472 </DiskSection>
2473 <!-- Describes all networks used in the package -->
2474 <NetworkSection>
2475     <Info>List of logical networks used in the package</Info>
2476     <Network ovf:name="VM Network">
2477         <Description>The network that the VMs connect to</Description>
2478         <!-- Network port profile for storage traffic -->
2479
2480         <NetworkPortProfileURI>http://www.dmtf.org/networkportprofiles/networkportprofile1.xml</Netwo
2481 rkPortProfileURI>
2482         <!-- Network port profile for networking traffic -->
2483
2484         <NetworkPortProfileURI>http://www.dmtf.org/networkportprofiles/networkportprofile2.xml</Netwo
2485 rkPortProfileURI>
2486     </Network>
2487 </NetworkSection>
2488 <VirtualSystemCollection ovf:id="vsc1">
2489     <Info>Collection of 2 VMs</Info>
2490     <VirtualSystem ovf:id="storage server">
2491         <Info>Describes a virtual machine</Info>
2492         <Name>Virtual Appliance One</Name>
2493         <ProductSection>
2494             <Info>Describes product information for the appliance</Info>
2495             <Product>The Great Appliance</Product>
2496             <Vendor>Some Great Corporation</Vendor>
2497             <Version>13.00</Version>

```

```

2498     <FullVersion>13.00-b5</FullVersion>
2499     <ProductUrl>http://www.somegreatcorporation.com/greatappliance</ProductUrl>
2500     <VendorUrl>http://www.somegreatcorporation.com/</VendorUrl>
2501     <Property ovf:key="admin.email" ovf:type="string">
2502         <Description>Email address of administrator</Description>
2503     </Property>
2504     <Property ovf:key="app.ip" ovf:type="string" ovf:defaultValue="192.168.0.10">
2505         <Description>The IP address of this appliance</Description>
2506     </Property>
2507 </ProductSection>
2508 <AnnotationSection ovf:required="false">
2509     <Info>A random annotation on this service. It can be ignored</Info>
2510     <Annotation>Contact customer support if you have any problems</Annotation>
2511 </AnnotationSection>
2512 <EulaSection>
2513     <Info>License information for the appliance</Info>
2514     <License>Insert your favorite license here</License>
2515 </EulaSection>
2516 <VirtualHardwareSection>
2517     <Info>Memory = 4 GB, CPU = 1 GHz, Disk = 100 GB, 1 Ethernet nic</Info>
2518     <Item>
2519         <rasd:AllocationUnits>Hertz*10^9</rasd:AllocationUnits>
2520         <rasd:Description>Virtual CPU</rasd:Description>
2521         <rasd:ElementName>1 GHz virtual CPU</rasd:ElementName>
2522         <rasd:InstanceID>1</rasd:InstanceID>
2523         <rasd:Reservation>1</rasd:Reservation>
2524         <rasd:ResourceType>3</rasd:ResourceType>
2525         <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2526     </Item>
2527     <Item>
2528         <rasd:AllocationUnits>byte*2^30</rasd:AllocationUnits>
2529         <rasd:Description>Memory</rasd:Description>
2530         <rasd:ElementName>1 GByte of memory</rasd:ElementName>
2531         <rasd:InstanceID>2</rasd:InstanceID>
2532         <rasd:ResourceType>4</rasd:ResourceType>
2533         <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2534     </Item>
2535     <EthernetPortItem>
2536         <epasd:Address>00-16-8B-DB-00-5E</epasd:Address>
2537         <epasd:Connection>VM Network</epasd:Connection>
2538         <epasd:Description>Virtual NIC</epasd:Description>
2539
2540         <epasd:ElementName>Ethernet Port</epasd:ElementName>
2541
2542         <epasd:InstanceID>3</epasd:InstanceID>
2543
2544         <epasd:NetworkPortProfileID>http://www.dmtf.org/networkportprofiles/networkportprofile1.xml</
2545     epasd:NetworkPortProfileID>
2546         <epasd:NetworkPortProfileIDType>2</epasd:NetworkPortProfileIDType>
2547         <epasd:ResourceType>10</epasd:ResourceType>
2548         <epasd:VirtualQuantityUnits>1</epasd:VirtualQuantityUnits>
2549     </EthernetPortItem>
2550     <StorageItem>
2551         <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
2552         <sasd:Description>Virtual Disk</sasd:Description>
2553         <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
2554         <sasd:InstanceID>4</sasd:InstanceID>
2555         <sasd:Reservation>100</sasd:Reservation>
2556         <sasd:ResourceType>31</sasd:ResourceType>
2557         <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
2558     </StorageItem>
2559 </VirtualHardwareSection>
2560 <OperatingSystemSection ovf:id="58" ovf:required="false">
2561     <Info>Guest Operating System</Info>
2562     <Description>OS</Description>
2563 </OperatingSystemSection>
2564 </VirtualSystem>
2565 <VirtualSystem ovf:id="web-server">
2566     <Info>Describes a virtual machine</Info>
2567     <Name>Virtual Appliance Two</Name>
2568 </ProductSection>

```

```

2569     <Info>Describes product information for the appliance</Info>
2570     <Product>The Great Appliance</Product>
2571     <Vendor>Some Great Corporation</Vendor>
2572     <Version>13.00</Version>
2573     <FullVersion>13.00-b5</FullVersion>
2574     <ProductUrl>http://www.somegreatcorporation.com/greatappliance</ProductUrl>
2575     <VendorUrl>http://www.somegreatcorporation.com/</VendorUrl>
2576     <Property ovf:key="admin.email" ovf:type="string">
2577         <Description>Email address of administrator</Description>
2578     </Property>
2579     <Property ovf:key="app.ip" ovf:type="string" ovf:defaultValue="192.168.0.10">
2580         <Description>The IP address of this appliance</Description>
2581     </Property>
2582 </ProductSection>
2583 <AnnotationSection ovf:required="false">
2584     <Info>A random annotation on this service. It can be ignored</Info>
2585     <Annotation>Contact customer support if you have any problems</Annotation>
2586 </AnnotationSection>
2587 <EulaSection>
2588     <Info>License information for the appliance</Info>
2589     <License>Insert your favorite license here</License>
2590 </EulaSection>
2591 <VirtualHardwareSection>
2592     <Info>Memory = 4 GB, CPU = 1 GHz, Disk = 100 GB, 1 Ethernet nic</Info>
2593     <Item>
2594         <rasd:AllocationUnits>Hertz*10^9</rasd:AllocationUnits>
2595         <rasd:Description>Virtual CPU</rasd:Description>
2596         <rasd:ElementName>1 GHz virtual CPU</rasd:ElementName>
2597         <rasd:InstanceID>1</rasd:InstanceID>
2598         <rasd:Reservation>1</rasd:Reservation>
2599         <rasd:ResourceType>3</rasd:ResourceType>
2600         <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2601     </Item>
2602     <Item>
2603         <rasd:AllocationUnits>byte*2^30</rasd:AllocationUnits>
2604         <rasd:Description>Memory</rasd:Description>
2605         <rasd:ElementName>1 GByte of memory</rasd:ElementName>
2606         <rasd:InstanceID>2</rasd:InstanceID>
2607         <rasd:ResourceType>4</rasd:ResourceType>
2608         <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
2609     </Item>
2610     <EthernetPortItem>
2611         <epasd:Address>00-16-8B-DB-00-5F</epasd:Address>
2612         <epasd:Connection>VM Network</epasd:Connection>
2613         <epasd:Description>Virtual NIC</epasd:Description>
2614
2615         <epasd:ElementName>Ethernet Port</epasd:ElementName>
2616         <!-- Virtual NIC for networking traffic -->
2617         <epasd:InstanceID>3</epasd:InstanceID>
2618
2619         <epasd:NetworkPortProfileID>http://www.dmtf.org/networkportprofiles/networkportprofile2.xml</
2620 epasd:NetworkPortProfileID>
2621         <epasd:NetworkPortProfileIDType>2</epasd:NetworkPortProfileIDType>
2622         <epasd:ResourceType>10</epasd:ResourceType>
2623         <epasd:VirtualQuantityUnits>1</epasd:VirtualQuantityUnits>
2624     </EthernetPortItem>
2625     <StorageItem>
2626         <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
2627         <sasd:Description>Virtual Disk</sasd:Description>
2628         <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
2629         <sasd:InstanceID>4</sasd:InstanceID>
2630         <sasd:Reservation>100</sasd:Reservation>
2631         <sasd:ResourceType>31</sasd:ResourceType>
2632         <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
2633     </StorageItem>
2634 </VirtualHardwareSection>
2635 <OperatingSystemSection ovf:id="58" ovf:required="false">
2636     <Info>Guest Operating System</Info>
2637     <Description>OS</Description>
2638 </OperatingSystemSection>
2639 </VirtualSystem>

```

```
2640     </VirtualSystemCollection>
2641 </Envelope>
```

2642 D.5 Example 5 (networkportprofile1.xml)

2643

2644 Network Port profile example for bandwidth reservation.

```
2645 <?xml version="1.0" encoding="UTF-8"?>
2646 <NetworkPortProfile xsi:schemaLocation="http://schemas.dmtf.org/ovf/networkportprofile/1
2647 http://schemas.dmtf.org/ovf/networkportprofile/1/dsp8049.xsd"
2648 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2649 xmlns="http://schemas.dmtf.org/ovf/networkportprofile/1"
2650 xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
2651 xmlns:epasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2652 schema/2/CIM_EthernetPortAllocationSettingData">
2653     <Item>
2654         <epasd:AllocationUnits>bit / second * 10^9</epasd:AllocationUnits>
2655         <epasd:ElementName>Network Port Profile 1</epasd:ElementName>
2656         <epasd:InstanceID>1</epasd:InstanceID>
2657         <epasd:NetworkPortProfileID>aaaaaaaa-bbbb-cccc-dddd-
2658 eeeeeeeeeeee</epasd:NetworkPortProfileID>
2659         <epasd:NetworkPortProfileIDType>3</epasd:NetworkPortProfileIDType>
2660         <epasd:Reservation>1</epasd:Reservation>
2661     </Item>
2662 </NetworkPortProfile>
```

2663 D.6 Example 6 (networkportprofile2.xml)

2664

2665 Network Port Profile example showing priority setting.

```
2666 <?xml version="1.0" encoding="UTF-8"?>
2667 <NetworkPortProfile xsi:schemaLocation="http://schemas.dmtf.org/ovf/networkportprofile/1
2668 http://schemas.dmtf.org/ovf/networkportprofile/1/dsp8049.xsd"
2669 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2670 xmlns="http://schemas.dmtf.org/ovf/networkportprofile/1"
2671 xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData"
2672 xmlns:epasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
2673 schema/2/CIM_EthernetPortAllocationSettingData">
2674     <Item>
2675         <epasd:AllowedPriorities>0</epasd:AllowedPriorities>
2676         <epasd:AllowedPriorities>1</epasd:AllowedPriorities>
2677         <epasd:DefaultPriority>0</epasd:DefaultPriority>
2678         <epasd:ElementName>Network Port Profile 2</epasd:ElementName>
2679         <epasd:InstanceID>2</epasd:InstanceID>
2680         <epasd:NetworkPortProfileID>aaaaaaaa-bbbb-cccc-dddd-
2681 ffffffff</epasd:NetworkPortProfileID>
2682         <epasd:NetworkPortProfileIDType>3</epasd:NetworkPortProfileIDType>
2683     </Item>
2684 </NetworkPortProfile>
2685
```

**ANNEX E
(informative)****Change Log**2686
2687
2688
2689

Version	Date	Description
1.0.0	2009-02-22	
1.1.0	2010-01-12	DMTF Standard release
2.0.0	2012-12-13	DMTF Standard release

2690

Bibliography

2691

2692 ISO 9660, *Joliet Extensions Specification*, May 1995,
2693 <http://littlesvr.ca/isomaster/resources/JolietSpecification.html>

2694 W3C, *Best Practices for XML Internationalization*, October 2008,
2695 <http://www.w3.org/TR/2008/NOTE-xml-i18n-bp-20080213/>

2696 DMTF DSP1044, *Processor Device Resource Virtualization Profile 1.0*
2697 http://www.dmtf.org/standards/published_documents/DSP1044_1.0.pdf

2698 DMTF DSP1045, *Memory Resource Virtualization Profile 1.0*
2699 http://www.dmtf.org/standards/published_documents/DSP1045_1.0.pdf

2700 DMTF DSP1047, *Storage Resource Virtualization Profile 1.0*
2701 http://www.dmtf.org/standards/published_documents/DSP1047_1.0.pdf

2702 DMTF DSP1022, *CPU Profile 1.0*,
2703 http://www.dmtf.org/standards/published_documents/DSP1022_1.0.pdf

2704 DMTF DSP1026, *System Memory Profile 1.0*,
2705 http://www.dmtf.org/standards/published_documents/DSP1026_1.0.pdf

2706 DMTF DSP1014, *Ethernet Port Profile 1.0*,
2707 http://www.dmtf.org/standards/published_documents/DSP1014_1.0.pdf

2708 DMTF DSP1050, *Ethernet Port Resource Virtualization Profile 1.1*
2709 http://www.dmtf.org/standards/published_documents/DSP1050_1.1.pdf

2710 DMTF DSP8049, *Network Port Profile XML Schema 1.0*
2711 http://schema.dmtf.org/ovf/networkportprofile/1/DSP8049_1.0.xsd
2712