



1
2
3
4

Document Number: DSP0265

Date: 2013-06-27

Version: 1.0.0

5 **Profile to Enable Automated Deployment of OVF**
6 **Packages**

7 **Document Type: Specification**

8 **Document Status: DMTF Standard**

9 **Document Language: en-US**

10

11 Copyright Notice

12 Copyright © 2012-2013 Distributed Management Task Force, Inc. (DMTF). All rights reserved.

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

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27 withdrawn or modified after publication, and shall be indemnified and held harmless by any party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

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

33

CONTENTS

34	Foreword	5
35	Introduction.....	7
36	1. Scope	9
37	1.1 Metadata Structure	9
38	1.2 The Meaning of Automated Deployment	9
39	2 Normative References.....	12
40	2.1 Approved References	12
41	2.2 Other References.....	12
42	3 Terms and Definitions	12
43	4 Abbreviated Terms	14
44	5 Synopsis	14
45	6 Description (informative)	14
46	6.1 Problem Area	14
47	6.2 Different Kinds of Incomplete Information	15
48	6.3 Information Needed for Deployment	15
49	7 Implementation.....	16
50	8 Methods.....	16
51	9 Use Cases	16
52	9.1 Deploy Copies of Virtual System Containing Guest Software Only	17
53	9.2 Deploy LAMP Server in a Single Virtual System	17
54	9.3 Deploy LAMP Server in Separate Virtual Systems.....	18
55	10 OVF Requirements.....	18
56	10.1 Mandatory Sections	19
57	10.2 Mandatory Elements	20
58	10.3 Naming Properties	22
59	10.4 Conditionally Mandatory Complete Property Groups	22
60	10.5 Properties for Multiple Instances of Network Interfaces	24
61	11 Conformance.....	25
62	11.1 Citation in OVF Envelope Element	25
63	11.2 XML Namespace.....	25
64	ANNEX A (informative) Use Cases for Future Consideration	26
65	ANNEX B (informative) Change Log.....	27
66		

67 Figures

68	Figure 1 – OVF Package Lifecycle	11
69		

70 Tables

71	Table 1 – OVF Sections Required.....	19
72	Table 2 – OVF Elements Required	20
73	Table 3 – OVF RASD Elements Required	21
74	Table 4 – OVF RASD Element Properties Required.....	21

75 Table 5 – Groups of Related Properties Network Address 23
76 Table 6 – Groups of Related Properties for Email Contact 23
77

78

Foreword

79 The Profile to Enable Automated Deployment of OVF Packages 1.0.0 (DSP0265) was prepared by the
80 SVPC OVF Working Group.

81 DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems
82 management and interoperability. For information about the DMTF, see <http://www.dmtf.org>.

83 Acknowledgments

84 The DMTF acknowledges the following individuals for their contributions to this document:

85 Editors:

86 Richard Landau - DMTF Fellow

87 Lawrence Lamers -VMware Inc.

88

89 Contributors

90 Hemal Shah - Broadcom

91 John Crandall - Brocade Communications Systems

92 Marvin Waschke - CA Technologies

93 Shishir Pardikar - Citrix Systems Inc.

94 Eric Wells - Hitachi, Ltd.

95 Robert Freund - Hitachi, Ltd.

96 HengLiang Zhang - Huawei

97 Jeff Wheeler - Huawei

98 Abdellatif Touimi - Huawei

99 Andreas Maier - IBM

100 Ron Doyle - IBM

101 John Leung - Intel Corporation

102 Cheng Wei - Microsoft Corporation

103 John Parchem - Microsoft Corporation

104 Monica Martin - Microsoft Corporation

105 Maurizio Carta - Microsoft Corporation

106 Narayan Venkat - NetApp

107 Srinivas Maturi - Oracle

108 Tatyana Bagerman - Oracle

109 Miguel Peñalvov - Telefónica

110 Steffen Grarup - VMware Inc.

111 Bhumip Khasnabish - ZTE Corporation

112 Junsheng Chu - ZTE Corporation

113 Ali, Ghazanfar - ZTE Corporation

114

116

Introduction

117 In order to promote the wide spread adoption of OVF it is important that software vendors have
118 confidence in the ability to build an OVF that can be deployed on a set of target virtualization platforms
119 (aka hypervisors). To this end it is useful to define additional constraints and requirements on the OVF
120 package to enable automated deployment and portability. Interoperability, i.e., the ability to be deployed
121 on target virtualization platforms, is also enhanced.

122 The Open Virtualization Format standard defines conformance requirements, but these are not sufficient
123 for the use cases that this specification addresses. Conformance can be done by inspection, checking for
124 the `ovf:required` tag in the OVF and noting the conformance level as specified in the standard.

125 Software developers need guidelines for what needs to be included in each section of the environment
126 file to ensure that a deployment function is capable of deploying the OVF.

127

128

129

130

131

132 Profile to Enable Automated Deployment of OVF Packages

133 1. Scope

134 1.1 Metadata Structure

135 OVF provides a metadata structure in which virtual machine appliance builders and packagers can
136 express requirements for successfully deploying a virtual system. The OVF Envelope is described in an
137 XML schema and a specification document that list the various elements of the metadata, their syntax,
138 and their semantics.

139 The OVF Envelope is a very general data structure that can express a variety of requirements of virtual
140 systems and their components. Because of the generality of the approach, and because of the wide
141 assortment of virtual systems to which it is intended to apply, there are few rules requiring the inclusion of
142 the many elements in the OVF spec. It is possible for an OVF Envelope to be syntactically valid according
143 to the OVF specification and schema, but yet not be useful for a deployment function attempting an
144 automated deployment.

145 To suit the needs of a particular application area, a profile can state requirements for the structure and
146 inclusion of metadata in an OVF package.

- 147 • This profile applies to a specific application area, namely, the automated deployment of virtual
148 systems contained in OVF packages for business and scientific applications in datacenters.
- 149 • This profile states conformance rules for OVF v1.1 Envelope files intended to be applied in this
150 area, That is, if an OVF package is intended to be used in an environment where it will be
151 deployed by automated mechanisms, with little or no human intervention, then the Envelope of
152 the package should conform to this profile.

153 This profile does not place restrictions on any mechanisms that may be used to deploy an OVF package.
154 It states criteria for an OVF package to *enable* a deployment function to be automated.

155 The scope of this specification of most interest pertains to the virtualization platforms for X86 architecture
156 processor systems. It can be applied to other processor architectures, however the capabilities of those
157 environments is beyond the scope of this specification.

158 Furthermore, the scope is aimed at software developers who author an OVF package for the explicit
159 purpose of enabling automated deployment. The author function identifies the target virtualization
160 platforms and the configuration requirements of the virtual machines and the environment they need for
161 operation and incorporates that into the OVF package.

162 Automated deployment allows for programmatic deployment of an OVF package (i.e., deployment without
163 input from a system administrator). Deployment data regarding policies and property values for
164 configuration may be supplied to augment the OVF package via programmatic means. That external
165 configuration data may have required human input. In addition the deployment function may apply policy
166 based on contractual agreements and environment considerations.

167 1.2 The Meaning of Automated Deployment

168 The goal of automated deployment is to minimize the human interaction required at the moment of
169 deployment of a virtual system. Ideally, the act of deploying a virtual machine might be a one-button
170 process: push the button and an instance of a virtual machine comes to life. This may be achievable to

171 greater or lesser degrees in various datacenter environments. The goal of this specification is to increase
172 the degree of automation that can be achieved by minimizing the intervention of a human system
173 manager at the time of deployment.

174 Deploying a virtual machine requires not only finding a suitable available virtual machine slot in a
175 managed virtual machine environment. It also requires tying the running virtual machine into the local
176 network and service pools, giving it a name and address so that clients can find it, connecting it to
177 databases that it needs, connecting it to sibling and partner processes, load balancers, and so forth.

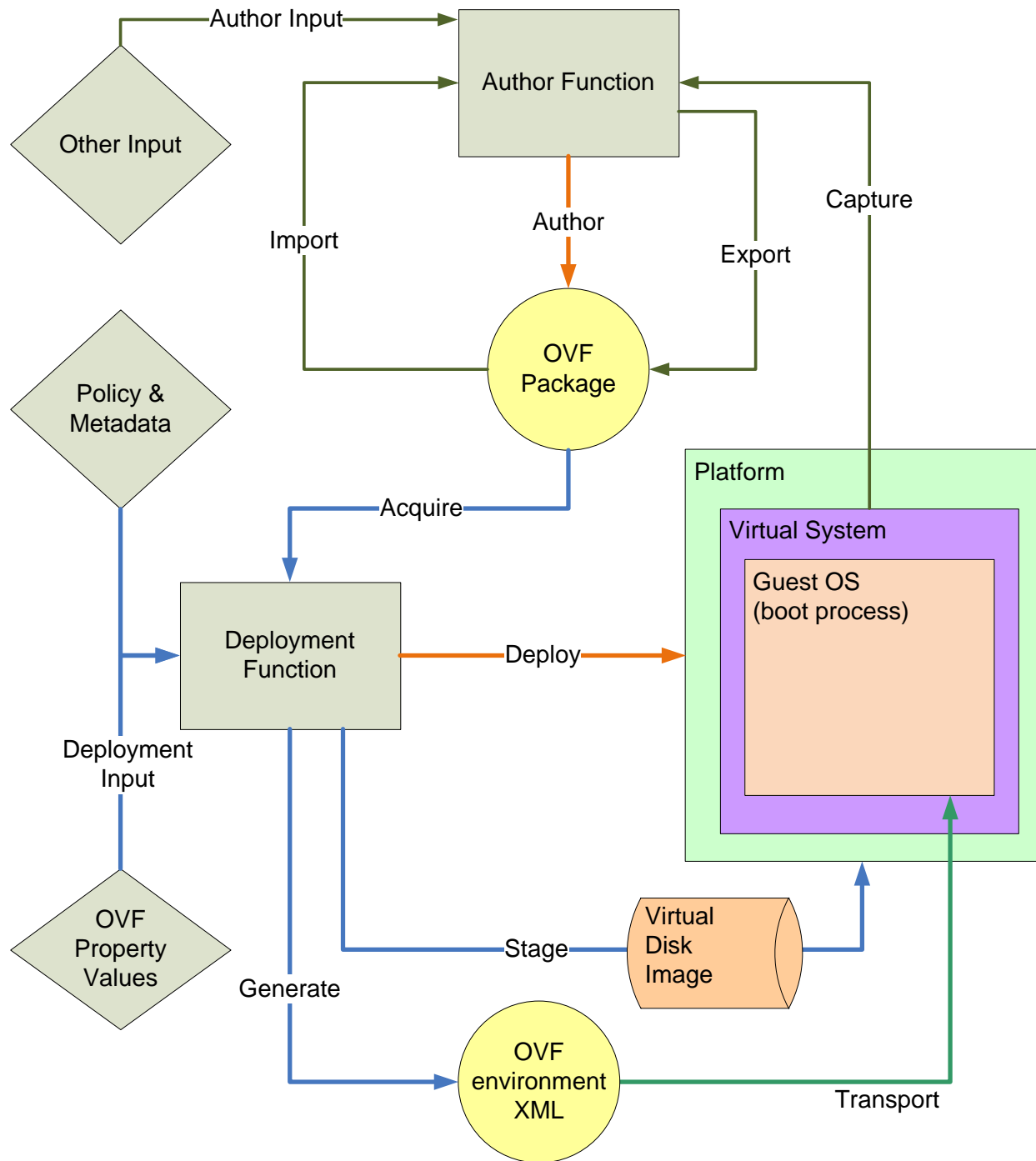
178 A deployment function can find a suitable virtual machine slot for a system only if it knows in advance
179 about the type of system required by the virtual machine, processors needed, memory, peripherals, and
180 such. The OVF Envelope of a virtual machine can supply this type of information. Additionally, a
181 deployment function can make such necessary connections to other local resources only if it is warned --
182 in advance, in writing, machine-readable -- about the needs of the virtual machine for connections to local
183 resources.

184 For instance, networking is the most obvious local resource needed, but the need may go beyond simply
185 an IP address. If a virtual machine is to be located by clients, then its address must be published in a way
186 that is accessible to them, e.g., in a dynamic DNS service. Or the virtual machine may need to be
187 connected to a load balancer if it is one of a scale-out group of systems. Or the virtual machine may need
188 access to an outgoing email service to send alarm or other messages.

189 A network connection is usually not simply a generic endpoint. Connections may be public or private, high
190 or low bandwidth, require firewalling or not, be named or anonymous, be shared behind a load balancer,
191 and so forth. If a deployment function knows the requirements of the network connection(s) of a virtual
192 machine, then it can sensibly allocate the right addresses and make other required connections.
193 (Example: a virtual machine does not know if it is being deployed as part of a scale-out set behind a load
194 balancer. But the load balancer certainly needs to be told about the new virtual machine. The deployment
195 function must do this at deployment time, ideally without human intervention.)

196 The interaction between the deployment function and the virtual machine at startup time must convey this
197 type of information. In a virtual machine described by an OVF, we assume that most or all of the
198 interaction will occur via the OVF Environment file supplied by the deployment function to the virtual
199 machine at boot time. For the deployment function to be able to do that, it needs to have had a list in
200 advance of the required resources and connections. Some data satisfying the requirements might be
201 statically assigned when the virtual machine is first installed into the deployment function's environment
202 (example: system name, if only one copy of the virtual machine is ever to be run). Other data might be
203 allocated from pools of resources established in advance by the system manager (example: IP
204 addresses). As a last resort, some data can be obtained from the system manager by interview at the
205 time of deployment. However, to maximize the automation of a deployment, we need to minimize the
206 interview. The intention of this specification is to encourage automated interaction between the author of
207 a virtual system, who knows what the system needs, and the system manager who is going to use a
208 particular virtual machine to deploy the virtual machine when the time comes.

209 Figure 1 illustrates the OVF package life cycle and defines verbs that describe actions taken. These
 210 terms are used in this specification.



211

212

Figure 1 – OVF Package Lifecycle

213 2 Normative References

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

217 2.1 Approved References

218 DMTF DSP0243, Open Virtualization Format Specification 1.1
219 http://www.dmtf.org/sites/default/files/standards/documents/DSP0243_1.1.0.pdf

220 DMTF DSP8023, OVF Envelope XSD 1.1.0
221 http://schemas.dmtf.org/ovf/envelope/1/dsp8023_1.1.xsd

222 DMTF DSP8027, OVF Environment XSD 1.1.0
223 http://schemas.dmtf.org/ovf/environment/1/dsp8027_1.1.xsd

224 DMTF CIM_ResourceAllocationSettingData.mof 2.22, included in CIM Schema v2.27
225 http://dmtf.org/standards/cim/cim_schema_v2270

226 DMTF CIM Schema 2.34.
227 http://dmtf.org/standards/cim/cim_schema_v2340

228 2.2 Other References

229 ISO/IEC Directives, Part 2, *Rules for the structure and drafting of International Standards*
230 <http://isotc.iso.org/livelink/livelink.exe?func=ll&objId=4230456&objAction=browse&sort=subtype>

231 3 Terms and Definitions

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

233 3.1

234 **can**

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

236 3.2

237 **cannot**

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

239 3.3

240 **conditional**

241 indicates requirements to be followed strictly in order to conform to the document when the specified
242 conditions are met

243 3.4

244 **Interoperability**

245 ISO/TR 16056-1:2004, 3.42

246 The ability of two or more systems (computers, communication devices, networks, software, and other
247 information technology components) to interact with one another and exchange information according to
248 a prescribed method in order to achieve predictable results.

- 249 **3.5**
250 **mandatory**
251 indicates requirements to be followed strictly in order to conform to the document and from which no
252 deviation is permitted
- 253 **3.6**
254 **may**
255 indicates a course of action permissible within the limits of the document
- 256 **3.7**
257 **need not**
258 indicates a course of action permissible within the limits of the document
- 259 **3.8**
260 **optional**
261 indicates a course of action permissible within the limits of the document
- 262 **3.9**
263 **OVF Portability**
264 The ability to export a virtual system from one author function to be imported into another author function.
- 265 **3.10**
266 **referencing profile**
267 indicates a profile that owns the definition of this class and can include a reference to this profile in its
268 "Related Profiles" table
- 269 **3.11**
270 **shall**
271 indicates requirements to be followed strictly in order to conform to the document and from which no
272 deviation is permitted
- 273 **3.12**
274 **shall not**
275 indicates requirements to be followed strictly in order to conform to the document and from which no
276 deviation is permitted
- 277 **3.13**
278 **should**
279 indicates that among several possibilities, one is recommended as particularly suitable, without
280 mentioning or excluding others, or that a certain course of action is preferred but not necessarily required
- 281 **3.14**
282 **should not**
283 indicates that a certain possibility or course of action is deprecated but not prohibited
- 284 **3.15**
285 **unspecified**
286 indicates that this profile does not define any constraints for the referenced CIM element or operation
- 287 **3.16**
288 **Workload**
289 A defined set of operations for the deployed ovf package to perform that validates the functionality and
290 that is visible in some fashion.

291 4 Abbreviated Terms

292 The following abbreviations are used in this document.

293 4.1

294 EPASD

295 CIM_EthernetPortAllocationSettingData that represents settings specifically related to allocation of an
296 Ethernet port resource.

297 4.2

298 LAMP

299 A Linux, Apache, MySQL and PHP (aka LAMP) OVF package that illustrates a multiple virtual machine
300 deployment.

301 4.3

302 OVF

303 Open Virtualization Format

304 4.4

305 RASD

306 CIM_ResourceAllocationSettingData that represents settings specifically related to an allocated resource.

307 5 Synopsis

308 **Profile Name:** Profile to Enable Automated Deployment of OVF Packages

309 **Version:** 1.0.0

310 **Organization:** SVPC OVF Working Group

311 **CIM Schema Version:** 2.29

312 **Central Class:** n/a

313 **Scoping Class:** n/a

314 **OVF Specification Version:** DSP0243 v1.1

315 6 Description (informative)

316 This profile describes requirements for the inclusion of metadata items in an OVF envelope. The
317 envelope may describe one or more virtual systems in an OVF package.

318 This profile is based on the specification and schema for OVF, specifically the OVF Envelope of an OVF
319 package. It states additional restrictions of mandatory and conditional structure of an Envelope to meet
320 the needs of an application area, the automated deployment of OVF packages.

321 6.1 Problem Area

322 This profile is targeted at deployment of virtual systems in medium businesses to large enterprises. The
323 virtual systems will be business, academic, or scientific appliances working in a network environment.

324 This profile is not targeted at small, one-on-one applications that do not use a network as a fundamental
325 part of the application.

326 If a virtualization platform is to deploy a virtual system with little or no immediate human assistance, it
327 must have metadata that answers many questions about the virtual system's needs and configuration. In
328 OVF packages, the metadata describing the virtual system(s) comes in the OVF Envelope file. That data
329 needs to be reasonably complete and comprehensive so that the deployment function does not need the
330 system manager to fill in many of the blanks.

331 As the Envelope is specified, most items within it are optional. This profile states requirements of
332 mandatory metadata items to be complete for this application area. Following these guidelines a builder
333 or packager of virtual systems can produce packages that can be deployed with a minimum of real-time
334 human interaction.

335 Many elements can be specified in an OVF envelope, but they are not required to be. Some of those
336 elements are important for automated deployment. Following the OVF schema and specification alone,
337 an envelope can be valid but uninformative. As an example, a responsible packager may include
338 declarations of the hardware used in the virtual system, but is not required to do so.

339 6.2 Different Kinds of Incomplete Information

340 It is possible for Envelope metadata to be incomplete in several different ways.

- 341 • An Envelope may neglect to describe all the hardware devices that are on the virtual system. Some
342 of the devices may need physical counterparts, in whole or in part, and thus must be declared to the
343 virtualization platform. For example, if the Guest Software is built for multiple cores, the envelope
344 must state the minimum number.
- 345 • An Envelope may describe some devices and configuration parameters incompletely, asking for
346 some information about an object but omitting critical items. For examples, requesting an IP address
347 for a network connection but not including the network mask or gateway or requesting an email
348 destination for problem messages, but not including the outbound email server address, account,
349 authentication information, etc.
- 350 • An Envelope may request some information but fail to request related information that is required for
351 successful deployment. For example, requesting properties in a product section but not specifying
352 the transport method for delivering the Environment file.

353 All of these types of incomplete or inconsistent information in an Envelope prevent a virtual systems from
354 being deployed in an automated way. The intent of this profile is to ensure that a compliant Envelope is
355 complete and consistent with respect to the information required for successful automated deployment of
356 the virtual system(s) that it describes.

357 6.3 Information Needed for Deployment

358 The deployment function and the virtual machine need to cooperate to accomplish a deployment with
359 minimal human intervention. The deployment function needs information about the structure and
360 requirements of the virtual machine to select an appropriate venue to run the virtual machine, and the
361 virtual machine needs information about the local environment to position itself correctly and
362 communicate with other resources.

363 Most of the information needed by the deployment function can be supplied in an OVF Envelope.

- 364 • Virtual system type, virtualized processor(s), memory required
- 365 • Virtual disk files required by the virtual machine, included in the package
- 366 • Peripheral devices to be virtualized
- 367 • Network attachments required
- 368 • Etc.

369 Information needed by the virtual machine at runtime, when it boots, must be supplied by the deployment
370 function and tailored to the local environment in which the virtual machine will run. This information will
371 come from a system manager, either from direct interview or from established policies. For example, IP
372 address might be assigned manually, or might come from pools established by management but
373 allocated by a DHCP server.

- 374 • System name and other identifying information
- 375 • Network addresses for network connections
- 376 • Locations of local resources and services, such as databases, email and other network services
- 377 • Etc.

378 To be reasonably portable across multiple deployment environments, a virtual machine must describe
379 itself and its resource requirements in a way that can be served by a deployment function in multiple
380 datacenter environments. For example, resource requirements should be as generic as possible, and not
381 depend on idiosyncratic behaviors of specific models of hardware or software.

382 7 Implementation

383 Not applicable. This clause intentionally left blank to preserve the numbering of sections.

384 8 Methods

385 Not applicable. This clause intentionally left blank to preserve the numbering of sections.

386 9 Use Cases

387 The following scenarios describe the needs of a user for metadata regarding virtual systems. In each
388 case, the user is attempting to install the virtual system(s) from an OVF package into a datacenter such
389 that they can be deployed later with no further need for human intervention.

390 The user's virtualization platform must have all the information that it will need in advance of an
391 automated deployment. In some cases, the information is required by the virtualization platform, and in
392 others the information will be required by the running virtual system when it is activated.

393 See ANNEX A for use cases that will be covered in a future release of this specification.

394 In these use cases, several different types of information may be needed by the deployment function
395 depending on the configuration of virtual systems. To enable automated deployment, this information
396 must be available to the deployment function in advance of the request to deploy one or more copies of a
397 virtual system.

- 398 • Information relating to the structure of the virtual system to be activated: the operating system,
399 instruction set, memory, CPU, disk, and peripheral requirements, and so forth. The deployment
400 function must choose a suitable virtual hardware system and provide the required virtual
401 hardware resources for the virtual system.
- 402 • Information that will make each system unique when multiple copies of the virtual system are
403 being run in the same naming domain, e.g., scaled-out in a load balanced set: system name, IP
404 address for management, other network addresses that are known globally, etc. To launch
405 multiple copies of a virtual system, the DA must be able to structure the copies to avoid naming
406 and addressing conflicts when all are active. If multiple copies are hidden behind a load balancer,
407 then the deployment function needs to assign predictable IP addresses (and names, if used) to

- 408 the load balancer and its subordinate virtual systems. The DA needs to understand in advance
409 what network connections will be used by virtual systems for what purposes.
- 410 • Information that enables multiple virtual systems to cooperate as a single application service:
411 e.g., names and addresses of partner and sibling components for a tiered application.
 - 412 ○ A web server virtual system (or multiple front-end web server virtual systems) need to
413 have access to the application virtual system (or virtual systems) for the business logic of
414 the application; to authentication and authorization servers or databases; and to a store
415 of web pages that may be managed by a separate virtual system.
 - 416 ○ An application virtual system needs to have access to the database of information being
417 served and to the store of application programs.
 - 418 ○ The addresses of the several components are not known in advance to the virtual
419 systems, but are supplied by the deployment function at the time of activation. To enable
420 automated deployment, the deployment function must know in advance the needs of
421 each virtual system, and must have control over delivering addresses to the virtual
422 systems. Again, to automate deployment with minimal human intervention, the
423 deployment function needs to understand in advance what network connections will be
424 used by virtual systems for what purposes.

425 9.1 Deploy Copies of Virtual System Containing Guest Software Only

426 The user wants to deploy copies of a virtual system for development, training, and testing. The virtual
427 system contains only an operating system built to a standard configuration.

428 Even for such skeletal virtual systems, a deployment function may need information about the
429 environment that the Guest Software expects, including:

- 430 • The family and specific version of the operating system,
- 431 • Hardware requirements of the Guest Software (CPUs, memory),
- 432 • Networks used by the Guest Software: the deployment function needs to know to which local
433 networks the virtual system needs to be connected. Some network connections may require special
434 treatment, such as dynamic address assignment, firewall permissions, and so forth.
- 435 • Scratch virtual disks used by the Guest Software,
- 436 • Other peripheral virtual devices to be used (e.g., CD drives, storage controllers)

437 9.2 Deploy LAMP Server in a Single Virtual System

438 The user wants to deploy copies of a LAMP server in a virtual system for development, training, and
439 testing, or as moderate-usage application servers. LAMP is a combination of standard components -- OS,
440 web server, database, and application language -- that can be used together in a tiered application. The
441 abbreviation "LAMP" refers to Linux, Apache, MySQL, and PHP (and possibly also Perl or Python).

442 The several software components in the LAMP stack in the virtual system run as separate processes in
443 the Guest Software and need to communicate with each other.

- 444 • Communication between the several processes of the LAMP server generally uses localhost IP.
445 Each process needs to know the (localhost) network addresses of one or more of its sibling
446 components. For instance, the web server needs to communicate with the application server and the
447 database; the application server needs to use the database process.
- 448 • Additionally, the application virtual system may need identity information and authentication
449 credentials to communicate with the database server, particularly if the database is external to the
450 database server, e.g., a pre-existing corporate database.

451 The web server needs IP address(es) assigned for external use. These may be fixed addresses assigned
452 by the manager and conveyed to the virtual system by the DA, or they may be dynamic addresses
453 assigned locally. The virtual system OVF needs to describe the number of addresses required and the
454 types of networks to which they must be attached.

455 **9.3 Deploy LAMP Server in Separate Virtual Systems**

456 The user wants to deploy a LAMP server as a collection of virtual systems. In this use case each
457 application is in a separate virtual system, i.e., a web server virtual system, an application server virtual
458 system, and a database server virtual system.

459 The virtual systems need to communicate with each other, therefore they need to know the network
460 addresses of other virtual systems.

- 461 • Since the components are packaged in separate virtual systems, and may run on separate
462 virtualization platforms, they need the network addresses of other components, and
463 authentication credentials, to function.
- 464 • The web server virtual system may need access to an authentication and authorization server or
465 database to validate user access before executing a transaction.
- 466 • The application virtual system may need identity information and authentication credentials to
467 communicate with the database server, particularly if the database is stored external to the
468 database server, e.g., a pre-existing corporate database.
- 469 • Any components that are scaled out into multiple copies need to be organized into a load
470 balancing set comprising a front end balancer with a known address and a set of server virtual
471 systems with hidden addresses known only to the balancer.

472 **10 OVF Requirements**

473 An OVF Envelope for automated deployment must contain sufficient information for a deployment
474 function to use without human intervention. A number of important sections and elements are needed that
475 are optional in the OVF spec and schema.

476 **10.1 Mandatory Sections**

477 The following table details the sections required within a compliant OVF Envelope.

478 **Table 1 – OVF Sections Required**

Element	Requirement	Description of Requirement
Sections Within the Envelope Container		
Envelope	Mandatory	An OVF Envelope shall contain at least one VirtualSystem section. That section may be contained in a VirtualSystemCollection.
Envelope	Mandatory	An OVF Envelope shall contain one DiskSection.
Envelope	Mandatory	An OVF Envelope shall contain one NetworkSection.
Envelope	Mandatory	An Envelope shall contain one ProductSection for each software component that is to be activated when the virtual system boots.
VirtualSystemCollection	Mandatory	If a VirtualSystemCollection section exists it shall contain at least one VirtualSystem section.
VirtualSystem	Mandatory	A VirtualSystem section shall contain one OperatingSystemSection.
VirtualSystem	Mandatory	A VirtualSystem section shall contain one VirtualHardwareSection.
VirtualHardwareSection	Mandatory	A VirtualHardwareSection shall contain one System section.

479

480 **10.2 Mandatory Elements**

481 The following table details elements required within particular sections of a compliant OVF Envelope.

482 **Table 2 – OVF Elements Required**

Element	Requirement	Description of Requirement
Other Elements Required Within Sections		
ProductSection	Mandatory	A ProductSection shall contain a Version element.
VirtualSystem	Mandatory	A VirtualSystem section shall contain an OperatingSystemSection element.
VirtualSystemCollection	Mandatory	A VirtualSystemCollection element shall contain at least one VirtualSystem section.
VirtualSystem	Mandatory	A VirtualSystem section shall contain a VirtualHardwareType element.
VirtualHardwareSection	Mandatory	If there are any Property requests in the Envelope, then the VirtualHardwareSection shall contain an ovf:transport attribute.
VirtualHardwareSection	Recommended	A VirtualHardwareSection should contain a System element, and the System element should contain a VirtualSystemType element that conforms to the CIM_VirtualSystemSettingData.VirtualSystemType property

483 OVF packages declare the virtual hardware requirements of the virtual system so that a hypervisor can
 484 present suitable virtual devices. Virtual systems and their hypervisors differ in the way that some
 485 peripheral devices are virtualized, particularly block I/O devices such as magnetic and optical disks. To
 486 maximize interoperability, OVF packages should include virtual system types that are well understood
 487 and documented in open standards.

488 In OVF, the vssd:VirtualSystemType element can be used to specify the virtualization platform that the
 489 virtual system is authored for. See CIM_VirtualSystemSettingData.VirtualSystemType property for
 490 description of how a vssd:VirtualSystemType is formatted. In some cases, the detailed VirtualSystemType
 491 descriptions supersede the general requirements stated below.

492 If a VirtualHardwareSection contains a System element with a registered value of
 493 vssd:VirtualSystemType, then the section shall meet the requirements of its type.

494 The following table details CIM_ResourceAllocationSettingData (RASD) elements required within a
 495 VirtualHardwareSection of an OVF Envelope that conforms to this standard.

496

Table 3 – OVF RASD Elements Required

RASD Resource Type Element	Requirement	Description of Requirement
RASD Elements Required, by ResourceType		
Processor	Mandatory	A VirtualHardwareSection shall contain at least one RASD element for the Processor.
Memory	Mandatory	A VirtualHardwareSection shall contain at least one RASD element for Memory.
IDE Controller, Parallel SCSI HBA, FC HBA, iSCSI HBA, IB HCA	Mandatory	A VirtualHardwareSection shall contain at least one RASD element representing a disk controller unless a vssd:VirtualSystemType is specified.
Ethernet Adapter	Mandatory	A VirtualHardwareSection shall contain at least one RASD element for Ethernet Adapter.
Ethernet Connection	Mandatory	A VirtualHardwareSection shall contain at least one RASD element for Ethernet Connection. If there are multiple Ethernet Adapter Items in a VirtualHardwareSection, then each Item shall contain at least one RASD element for Ethernet Connection.
CD Drive	Conditional	If the VirtualHardwareSection element specifies transport="iso" then the section should contain a RASD for a peripheral device capable of reading an ISO9660 conformant image.
Logical Disk	Mandatory	A VirtualHardwareSection shall contain at least one RASD element for a Logical Disk.

497 The following table details properties that are required to be declared on particular RASD elements of a
 498 compliant OVF Envelope.

499

Table 4 – OVF RASD Element Properties Required

Element	Requirement	Description of Requirement
RASD Element Properties Required, by ResourceType		
Processor	Mandatory	A Processor element shall contain an AllocationUnits and a Reservation property that specifies the compute capacity needed. A VirtualQuantity property may also be specified to indicate the number of virtual processors exposed to the Guest Software.
Memory	Mandatory	A Memory element shall contain an AllocationUnits and a Reservation property that specifies the minimum amount of memory needed.
Ethernet Adapter	Mandatory	An Ethernet Adapter RASD element shall contain a rasd:Connection property element.
Ethernet Connection	Mandatory	An Ethernet Connection RASD element shall contain a rasd:Connection property element.

500

501 10.3 Naming Properties

502 Many OVF Envelope files will include OVF Property declarations to retrieve configuration data from the
503 local environment. For the OVF keys of these Property declarations to be sensible to an automated
504 deployment function, the names and semantics of the keys must be known in advance.

505 For example, an arbitrary string such as "adminEmail" is not necessarily recognized by an automated
506 deployment function. Unless it is standardized and its semantics clearly defined, such a string is an
507 arbitrary choice of the OVF author. A human might understand the intention, but software may not.
508 Therefore, to ensure predictable and reliable interaction between an OVF package and an automated
509 deployment function, this specification must establish the OVF key names and semantics for commonly
510 accessed configuration information.

511 In many cases, the CIM Schema already contains a CIM property with semantics that match the needs of
512 an OVF Property. In this case, the name and the semantics of the existing CIM property will be used.
513 Reuse of existing, defined and vetted technology minimizes the effort required for integration with
514 management applications and eliminates confusion and version skew.

515 Where a CIM property serves the needs of an OVF Property, the name of the ovf:key to be used is
516 constructed as follows:

```
517 <CIM classname>.<CIM property name>
```

518 The combination of the CIM class name and property name within that class specifies unambiguously the
519 semantics desired for the OVF Property. Version compatibility rules of the CIM architecture ensure that
520 the semantics for an established name will not change incompatibly.

521 Where there is no current CIM property that meets the needs of the OVF Property declaration, this
522 specification will invent a name and semantics to be used. When the CIM Schema is updated to contain a
523 suitable CIM property, then the CIM name may be adopted in a minor version update of this specification,
524 and the OVF-invented name may become a deprecated but tolerated synonym for the CIM name.

525 10.4 Conditionally Mandatory Complete Property Groups

526 Some properties requested by the Envelope from the virtualization platform are not usable alone, but
527 naturally occur in groups. For example, a destination email address for some type of notification is not
528 usable without knowledge of an outbound SMTP server and sufficient credentials to use that server to
529 send mail.

530 The following table details the groups of properties that are required to be requested as groups in order to
531 provide sufficient information at runtime. These groups of properties are referenced to existing CIM
532 classes for clarity and interoperability. The data syntax and semantics of the properties values supplied in
533 the Environment file, in response to Property declarations in the Envelope, shall conform to the data
534 syntax and semantics of the corresponding properties in the classes specified here. See DSP0243_1.1.0
535 Product Section for additional requirements.

536 Some properties naturally occur in groups. For example, many early examples of OVF Envelopes called
537 for properties such as "adminEmail" intending to be given an address to which to send urgent email
538 notices to the system administrator. In the case of automated deployment, this property request is
539 insufficient in at least two ways.

540 A single email address is incomplete. A destination address alone is generally not sufficient for a virtual
541 system to send mail through SMTP or Exchange. The virtual system does not know where an upstream
542 SMTP service is located in the network and such mail services require authentication.

543 The groups proposed in Table 5 include information sufficient for the intended applications.

544

Table 5 – Groups of Related Properties Network Address

Group	Reference CIM Class	CIM Property
Boot Origin	CIM_IPAssignmentSettingData	AddressOrigin
IPv4 Address	CIM_StaticIPAssignmentSettingData	IPv4Address SubnetMask GatewayIPv4Address
IPv6 Address	CIM_StaticIPAssignmentSettingData	IPv6Address IPv6AddressType IPv6SubnetPrefixLength GatewayIPv6Address
Name Service	CIM_DNSSettingData	DNSServerAddresses[]

545 Note:

- 546 • Property keys and values are case sensitive.
- 547 • Property keys that end with a dot and number could be a list.
- 548 • Some keys may depend on others to have a meaning, e.g., network.ipaddr_IPv4 is meaningful if
549 network.bootproto is not 'dhcp'.

550

Table 6 – Groups of Related Properties for Email Contact

Group	Reference	Property
Contact	RFC6068	Email to address
SMTP Service Address	IP address plus RFC6409 for port	SMTPServiceAddress
Mail Service	CIM_Account	Name UserPassword
Authentication	See Annex	IsSPAAuthenticationRequired

551 Email To address
552 string ToAddress
553 String designating the intended recipient or recipients of the message. The string shall contain a
554 mailto URI as specified in RFC6068. This URI may contain multiple recipients and CC and BCC
555 options, within the limits specified by the RFC.

556 SMTP service address including port
557 string SMTPServiceAddress
558 String specifying the IP address and port of an SMTP service to be used to transmit the email. If the
559 string does not specify the port, then the default port specified in RFC6409 shall be used.

560 SMTP login name
561 string CIM_Account.Name
562 For an SMTP service that is listed in the CIM_Account.Host array of a CIM_Account, this shall be the
563 name of an account that is privileged to send email through the service.

564 SMTP login password
565 string CIM_Account.UserPassword
566 For a CIM_Account on an SMTP service that is listed in the CIM_Account.Host array of a
567 CIM_Account, this shall be the password associated with the CIM_Account.Name.

568 SMTP login requires SPA authentication
569 boolean SMTPIsSPAAuthenticationRequired
570 Boolean specifying whether SPA (Secure Password Authentication) is required to access the SMTP
571 service.

572 Editors Note:

573 We (the editors) believe that it is acceptable for a specialization, such as this spec, to employ or require
 574 elements from various versions of other specifications on which it depends. For example, an OVF
 575 Envelope file can assert conformance with this spec if it complies with OVF v1.1 (DSP0243) and includes
 576 elements from a CIM Schema version later than that required by the OVF spec. So long as the OVF
 577 Envelope file cites the correct XML namespaces for the CIM elements used, there is no fundamental
 578 conflict between versions. In particular, the preferred XML namespace names for CIM classes specify
 579 only the major version, and therefore imply the latest minor version of that major version; thus CIM
 580 properties and RASD values introduced in CIM versions after 2.22 will be available.

581 10.5 Properties for Multiple Instances of Network Interfaces

582 Some properties within a single ProductSection must occur multiple times. For example, when a virtual
 583 system has more than one Ethernet interface and the interfaces are to be attached to different networks,
 584 the OVF Envelope and the deployment function must cooperate to provide correct address and
 585 connection information for the several interfaces.

586 To continue the example, if two interfaces require addresses on separate networks, then the Property
 587 declarations in the Envelope, and the matching declarations in the Environment, must specify the network
 588 attachment of each interface. Note that, in the VirtualHardwareSection, the ovf:Item declarations of the
 589 two interfaces specify the network connections to be used in rasd:Connection elements. Therefore, each
 590 Property declaration shall refer to a specific ovf:Item declaration, and each ovf:Item declaration shall refer
 591 to its desired network connection.

592 Specifically, in the ovf:Envelope,

- 593 • Property declarations that reference multiple instances of an Ethernet interface within a
 594 ProductSection shall specify the corresponding ovf:Item in the ovf:VirtualHardwareSection; the
 595 Property declaration shall include a dsp265:rasdinstanceid attribute the value of which is the
 596 rasd:InstanceID value of the ovf:Item.
- 597 • ovf:Item declarations in the ovf:VirtualHardwareSection shall include a rasd:Connection item
 598 specifying the network to which the interface is to be attached.

599 Example:

600 For two Ethernet interfaces, the Envelope may contain the following declarations:

```
601 <Envelope xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1"
602 xmlns:dsp265="http://schemas.dmtf.org/ovf/dsp0265/1">
603
604 <!--
605 Request for information for two ethernet interfaces.
606
607 Note that the instance ids point to the corresponding RASD elements for the
608 interfaces; the RASD element rasd:Connection specifies which physical or
609 logical network the interface is attached to. The deployment function needs
610 to be able to distinguish the several (virtual) interfaces in order to
611 assign them addresses on their specific networks. To accomplish this,
612 a new attribute, dsp265:rasdinstanceid, contains the RASD InstanceID string
613 in both the Envelope request and the Environment reply.
614 -->
615
616 <ProductSection>
617 <Info></Info>
618 <Product></Product>
619 <Vendor></Vendor>
620 <Version></Version>
621 <Property ovf:key="CIM_StaticIPAssignmentSettingData.IPv4Address"
622 ovf:type="string" dsp265:rasdinstanceid="6">
```



```

623         <Label>Ethernet on virtual system Network</Label>
624         <Description>IPv4 addr of this virtual system on the virtual system
625 Network for normal external and internal data</Description>
626     </Property>
627     <Property ovf:key="CIM_StaticIPAssignmentSettingData.IPv4Address"
628 ovf:type="string" rasdinstanceid="7">
629         <Label>Ethernet on Admin Network</Label>
630         <Description>IPv4 addr of this virtual system for private traffic on the
631 administrative network</Description>
632     </Property>
633 </ProductSection>
634
635 </Envelope>

```

636 And the Environment constructed by the deployment function may contain responses such as the
637 following.

```

638 <Environment xmlns:ovfenvir="http://schemas.dmtf.org/ovf/environment/1"
639 xmlns:dsp265="http://schemas.dmtf.org/ovf/dsp0265/1">
640
641     <!--
642     Environment response to requests in the Envelope.
643     -->
644
645     <ProductSection>
646         <Property ovfenvir:key="CIM_StaticIPAssignmentSettingData.IPv4Address"
647 ovfenvir:type="string" dsp265:rasdinstanceid="6" ovfenvir:value="10.0.1.1">
648         </Property>
649         <Property ovfenvir:key="CIM_StaticIPAssignmentSettingData.IPv4Address"
650 ovfenvir:type="string" dsp265:rasdinstanceid="7" ovfenvir:value="10.99.1.1">
651         </Property>
652
653     </ProductSection>
654
655 </Environment>

```

657 Ethernet interfaces are special in this regard, that they are to be assigned to separate networks. (For
658 virtual systems, there is no point in teaming multiple virtual interfaces.) If other devices are found to
659 require external management of multiple instances, they will need to be treated similarly.

660 11 Conformance

661 11.1 Citation in OVF Envelope Element

662 To signify conformance with this profile, an OVF envelope file shall include the following citation as a
663 direct child of the ovf:Envelope element.

```

664 <ovfprofiles:ProfileSupported>
665     <ovfprofiles:ProfileName>Profile to Enable Automated Deployment of OVF
666 Packages</ovfprofiles:ProfileName>
667     <ovfprofiles:ProfileVersion>1.0</ovfprofiles:ProfileVersion>
668 </ovfprofiles:ProfileSupported>

```

669 11.2 XML Namespace

670 The <ovfprofiles:ProfileSupported> element shall be declared in the following XML namespace.

```

671 xmlns:ovfprofiles="http://schemas.dmtf.org/ovf/profiles/1"

```

ANNEX A (informative)

672
673
674
675
676

Use Cases for Future Consideration

677 The following use cases are not covered in this specification as they need extensions to OVF that are not
678 in the current release. When the next OVF update is released these will be addressed.

679 **A.1 Deploy General Tiered Application**

680 The user wants to deploy a tiered application as a collection of virtual systems. The tiered application may
681 include a web server, one or more application servers for the business logic of the application, and one or
682 more database servers.

683 As with the separate LAMP servers, some of the virtual systems need to communicate with others and
684 need to know network addresses and authentication credentials.

685 **A.2 Deploy Scaled-out Copies of Virtual System**

686 The user wants to deploy a number of scaled-out copies of an application behind a front-end load
687 balancer. All of the components are packaged as virtual systems.

- 688 • The load balancer is assigned an externally known network address.
- 689 • The scaled-out virtual systems may be assigned DHCP pool addresses.
- 690 • The load balancer needs to know the addresses of all the virtual systems in its group.
- 691 • If the application is tiered, the virtual systems may need identity and credentials to communicate with
692 components in other layers.
- 693 • It is also possible that the scaled-out virtual systems in the same layer need to communicate with
694 each other, for example, for locking certain resources.

695
696
697
698

ANNEX B
(informative)
Change Log

Version	Date	Description
1.0.0	2013-06-27	

699