



Open Virtualization Format White Paper

Version 1.0.0

Status: Informational

Publication Date: 2/6/2009

DSP2017

Copyright © 2009 Distributed Management Task Force, Inc. (DMTF). All rights reserved.

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

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patent owner or claimant, and shall have no liability or responsibility for costs or losses incurred if a standard is withdrawn or modified after publication, and shall be indemnified and held harmless by any party implementing the standard from any and all claims of infringement by a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

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

Open Virtualization Format White Paper

OVF version 1.0.0e

Version 1.0.0

Publication Date: 2/6/2009

DSP2017

Status: Informational

Abstract

This white paper describes the Open Virtualization Format (OVF). OVF is a hypervisor-neutral, efficient, extensible, and open specification for the packaging and distribution of virtual appliances composed of one or more virtual computer systems. The target audience of this white paper is anyone who wants to understand OVF and its reason for development. Some familiarity with virtualization and the general concepts of the CIM model is assumed.

1

Table of Contents

2	1 Introduction	5
3	1.1 Overview	5
4	1.2 Virtual Appliances	6
5	1.3 Design Goals	6
6	1.4 Virtual Appliance Life-Cycle	8
7	2 Portable Virtualization Format	9
8	2.1 OVF Package	9
9	2.2 OVF Environment	9
10	2.3 Sample OVF Descriptor	10
11	3 Using the Open Virtualization Format	11
12	3.1 Creation	11
13	3.2 Deployment	12
14	4 Features	13
15	4.1 Virtual Hardware Description	13
16	4.2 Deployment Options	14
17	4.3 Deployment Customization	15
18	4.4 Internationalization	15
19	4.5 Extensibility	16
20	4.6 Conformance	16
21	5 Portability	17
22	6 Future Versions of the OVF Specification	18
23	7 Conclusion	18
24	A Multi-tiered Petstore Example	20
25	Architecture and Packaging	20
26	Properties	20
27	Disk Layout	21
28	Complete OVF Descriptor	22
29	Complete OVF Environments	27
30	B LAMP Stack Example	29
31	Deployment-time Customization	29
32	Simple LAMP OVF Descriptor	30
33	Two-tier LAMP OVF Descriptor	33
34	C Extensibility Example	38
35	Custom Schema	38
36	Descriptor with custom extensions	39

37

38 1 Introduction

39 1.1 Overview

40 The rapid adoption of virtual infrastructure has highlighted the need for a standard, portable meta-data
41 model for the distribution of virtual machines to and between virtualization platforms. Packaging an
42 application together with the operating system on which it is certified, into a virtual machine that can be
43 easily transferred from an ISV, through test and development and into production as a pre-configured,
44 pre-packaged unit with no external dependencies, is extremely attractive. Such pre-deployed, ready to
45 run applications packaged as virtual machines (VMs) are called virtual appliances. In order to make this
46 concept practical on a large scale it is important that the industry adopts a vendor-neutral standard for the
47 packaging of such VMs and the meta-data that are required to automatically and securely install,
48 configure, and run the virtual appliance on any virtualization platform.

49 Virtual appliances are changing the software distribution paradigm because they allow application
50 builders to optimize the software stack for their application and deliver a turnkey software service to the
51 end user. For solution providers, building a virtual appliance is simpler and more cost effective than
52 building a hardware appliance, since the application is pre-packaged with the operating system that it
53 uses, reducing application/OS compatibility testing and certification, and allowing the software to be pre-
54 installed in the OS environment it will run in – by the ISV. For end users, virtual appliances offer an
55 opportunity to dramatically simplify the software management lifecycle through the adoption of a
56 standardized, automated, and efficient set of processes that replace OS and application specific
57 management tasks today.

58 Whereas current virtual appliances contain a single VM only, modern enterprise applications model
59 service oriented architectures (SOA) with multiple tiers, where each tier contains one or more machines.
60 A single VM model is thus not sufficient to distribute a multi-tier service. In addition, complex applications
61 require install-time customization of networks and other customer specific properties. Furthermore, a
62 virtual appliance is packaged in a run-time format with hard disk images and configuration data suitable
63 for a particular hypervisor. Run-time formats are optimized for execution and not for distribution. For
64 efficient software distribution, a number of additional features become critical, including portability,
65 platform independence, verification, signing, versioning, and licensing terms.

66 The Open Virtualization Format (OVF) specification is a hypervisor-neutral, efficient, extensible, and open
67 specification for the packaging and distribution of virtual appliances composed of one or more VMs. It
68 aims to facilitate the automated, secure management not only of virtual machines but the appliance as a
69 functional unit. For the OVF format to succeed it must be developed and endorsed by ISVs, virtual
70 appliance vendors, operating system vendors, as well as virtual platform vendors, and must be developed
71 within a standards-based framework.

72 This document gives a detailed description of the motivation and goals behind the design of OVF, and
73 should be read as an accompaniment to the OVF specification of the same revision number.

74

75 1.2 Virtual Appliances

76 A virtual appliance is a pre-configured software stack comprising one or more virtual machines. Each
77 virtual machine is an independently installable run-time entity comprising an operating system,
78 applications and other application-specific data, as well as a specification of the virtual hardware that is
79 required by the virtual machine. Many infrastructure applications and even end-user applications that are
80 accessible over a network, such as a DNS server, a bug tracking database, or a complete CRM solution
81 composed of a web, application and database tier, can be delivered as virtual appliances. Delivering
82 complex software systems and services as a pre-configured software stack can dramatically increase
83 robustness and simplify installation. Virtual appliances need not be developed and delivered by 3rd party
84 ISVs – the concept is equally useful and often used within an enterprise in which a *virtual machine*
85 *template* for a particular service is assembled, tested, and certified by an IT organization and then
86 packaged for repeated, “cookie cutter” deployment throughout the enterprise.

87 Commonly, a software service is implemented as a multi-tier application running in multiple virtual
88 machines and communicating across the network. Services are often composed of other services, which
89 themselves might be multi-tier applications or composed of other services. This is known as service-
90 oriented architecture or SOA. Indeed the SOA-type model naturally fits into a virtual appliance-based
91 infrastructure, since virtual appliances are typified by the use of network facing, XML based management
92 and service interfaces that allow composition of appliances to deliver a complete application.

93 For example, consider a typical web application that consists of three tiers. A web tier that implements the
94 presentation logic, and application server tier that implements the business logic, and a back-end
95 database tier. A straightforward implementation would divide this into 3 virtual machines, one for each
96 tier. In this way, the application can scale from the fraction of a single physical host to 3 physical hosts.
97 Another approach is to treat each tier as a service in itself. Hence, each tier is a multi-VM service that
98 provides a clustered solution. This can provide far greater scalability than just up to 3 physical hosts.
99 Taking the web-front example, a common scenario is to have many web servers, fewer applications
100 servers, and one or two database servers. Implemented as virtual machines, each tier can scale across
101 as many or as few physical machines as required, and each tier can support multiple instances of service
102 VMs.

103

104 1.3 Design Goals

105 The Open Virtualization Format (OVF) describes an open, secure, portable, efficient and extensible
106 format for the packaging and distribution of (collections of) virtual machines. The key properties of the
107 format are:

- 108 • **Optimized for distribution**
109 Supports content verification and integrity checking based on industry standard public key
110 infrastructure, and provides a basic scheme for management of software licensing.
- 111 • **Optimized for a simple, automated user experience**
112 Supports validation of the entire package and each virtual machine or meta-data component of
113 the OVF during the installation phases of the VM lifecycle management process. It also packages
114 with the appliance relevant user-readable descriptive information that can be use by a
115 virtualization platform to streamline the installation experience.
- 116 • **Supports both single VM and multi-VM configurations**
117 Supports both standard single VM packages, and packages containing complex, multi-tier
118 services consisting of multiple interdependent VMs.
- 119 • **Portable VM packaging**
120 OVF is virtualization platform neutral, while also enabling platform-specific enhancements to be
121 captured. It supports the full range of virtual hard disk formats used for VMs today, and is
122 extensible to deal with future formats that may arise. Virtual machine properties are captured
123 concisely and accurately.

- 124 • **Vendor and platform independent**
125 The OVF does not rely on the use of a specific host platform, virtualization platform, or guest
126 operating system (within the appliance).
- 127 • **Extensible**
128 OVF is immediately useful – and extensible. It is designed to be extended as the industry moves
129 forward with the virtual appliance technology. It also supports and permits the encoding of custom
130 meta-data to support specific vertical markets.
- 131 • **Localizable**
132 Supports user visible descriptions in multiple locales, and supports localization of the interactive
133 processes during installation of an appliance. This allows a single packaged appliance to serve
134 multiple market opportunities.
- 135 • **Open standard**
136 The OVF has arisen from the collaboration of key vendors in the industry, and will be developed
137 as a future standard for portable virtual machines.
138

139 From the user's point of view, an OVF is a **packaging format for software appliances**. Once installed,
140 an OVF adds to the user's infrastructure a self-contained, self-consistent, software solution for achieving
141 a particular goal. For example, an OVF might contain a fully-functional and tested web-server / database /
142 OS combination, such as a LAMP stack (Linux + Apache + MySQL + PHP), or it may contain a virus
143 checker, including its update software, spyware detector, etc.
144

145 From a technical point of view, an OVF is a **transport** mechanism for **virtual machine templates**. One
146 OVF may contain a single VM, or many VMs (it is left to the software appliance developer to decide which
147 arrangement best suits their application). OVFs must be installed before they can be run; a particular
148 virtualization platform may run the VM from the OVF, but this is not required. If this is done, the OVF itself
149 can no longer be viewed as a “golden image” version of the appliance, since run-time state for the virtual
150 machine(s) will pervade the OVF. Moreover the digital signature that allows the platform to check the
151 integrity of the OVF will be invalid.
152

153 As a transport mechanism, OVF differs from VMware's VMDK Virtual Disk Format and Microsoft's VHD
154 Virtual Hard Disk format or the open source QCOW format. These are run-time VM image formats,
155 operating at the scope of a single VM disk, and though they are frequently used as transport formats
156 today, they are not designed to solve the VM portability problem; they don't help you if you have a VM
157 with multiple disks, or multiple VMs, or need customization of the VM at install time, or if your VM is
158 intended to run on multiple virtualization platforms (even if the virtualization platforms claim support of the
159 particular virtual hard disk format used).
160

161 Included within the OVF remit is the concept of the **certification and integrity** of a packaged software
162 virtual appliance, allowing the platform to determine the provenance of the appliance, and to allow the
163 end-user to make the appropriate trust decisions. The OVF specification has been constructed so that
164 the appliance is responsible for its own configuration and modification. In particular, this means that the
165 virtualization platform does not need to be able to read from the appliance's file systems. This decoupling
166 of platform from appliance means that OVFs may be implemented using any operating system, and
167 installed on any virtualization platform that supports the OVF format. A specific mechanism is provided for
168 appliances to detect the platform on which they are installed, and react to it. This allows platforms to
169 extend this specification in unique ways without breaking compatibility of appliances across the industry.
170

171 The OVF format has several specific features that are designed for complex, multi-tier services and their
172 associated distribution, installation, configuration and execution workflow:

- 173 • It directly supports the configuration of multi-tier applications and the composition of virtual
174 machines to deliver composed services.
- 175 • It permits the specification of both VM and application-level configuration.

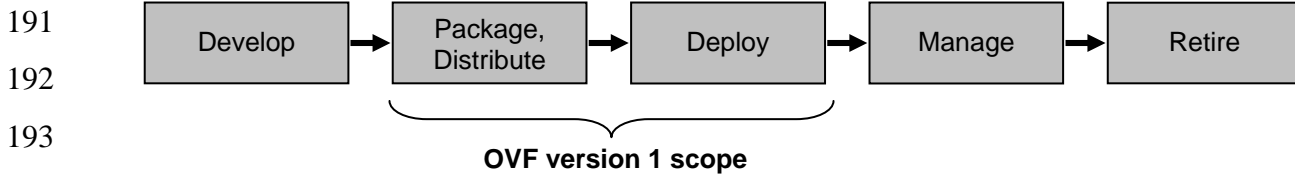
- 176 • It offers robust mechanisms for validation of the contents of the OVF, and full support for
177 unattended installation to ease the burden of deployment for users, and thereby enhance the
178 user's experience.
- 179 • It uses commercially accepted procedures for integrity checking of the contents of the OVF,
180 through the use of signatures and trusted third parties. This serves to reassure the consumer of
181 an appliance that it has not been modified since signed by the creator of the appliance. This is
182 seen as critical to the success of the virtual appliance market, and to the viability of independent
183 creation and online download of appliances.
- 184 • It permits commercial interests of the appliance vendor and user to be respected, by providing a
185 basic method for presentation and acknowledgement of licensing terms associated with the
186 appliance.

187

188 1.4 Virtual Appliance Life-Cycle

189 The software life cycle for virtual appliances is shown below:

190



195 A service, consisting of one or more VMs and the relevant configuration and deployment meta data, is
196 packaged into the OVF format at the end of the development phase. The components used here can be
197 third-party components. For example, a clustered database component might be acquired from a third-
198 party ISV. The deployment phase is the installation of an OVF package. The management and retirement
199 phase is specific to the virtualization product used, and to the contents of the OVF itself. Management
200 includes, for example, ongoing maintenance and upgrade of the appliance, which is likely to be highly
201 dependent on the contents of the VMs in the OVF. In the retirement phase, the software is
202 decommissioned and any resources it consumes are released. In this version of the OVF specification we
203 deal specifically with the packaging, distribution and deployment phases. Later versions of the
204 specification may address management and retirement in detail.

205

206

207 **2 Portable Virtualization Format**

208 The Open Virtualization Format defines a format for distributing software to be deployed in virtual
209 machines, and an environment for which they execute. This is respectively known as the OVF package
210 and the OVF environment.

211 **2.1 OVF Package**

212 The OVF package consists of an OVF descriptor and a set of additional content, typically virtual disks.
213 Content can accompany the package directly or be referred externally via HTTP. The specification also
214 enables an entire OVF package to be distributed as a single file.

215 The OVF descriptor is an XML document that describes meta-data about the software installed on the
216 virtual disks. The OVF specification 1.0 specification defines the common sections used for deploying
217 software efficiently, such as virtual hardware, disks, networks, resource requirements, and customization
218 parameters. The descriptor is designed to be extensible so further information can be added later.

219 The specification allows any virtual disk format to be used, as long as the disk format specification is
220 public and without restrictions. This supports the full range of virtual hard disk formats used for
221 hypervisors today, and it is extensible to allow for future formats.

222 The virtual disk format will commonly be some simple basic disk block format agnostic to the guest OS
223 installed. By way of example, VMware VMDK formats deal with 512 byte disk sectors stored in 64KB
224 blocks, in a number of flat, sparse, and compressed variants. At deployment time, the virtualization
225 platform creates virtual disks in a basic disk block format it prefers. The runtime virtual disk format may
226 be identical to the distribution format, but will often be different; it may for instance not be efficient to run
227 out of a compressed virtual disk format. Finally, the guest OS installed on the virtual disk has its own disk
228 file format, such as NTFS, EXT3, or ZFS, but this is not relevant to describe or understand at the OVF
229 level.

230 See section 2.3 and appendix A and B for examples of OVF descriptors.

231

232 **2.2 OVF Environment**

233 A virtual appliance often needs to be customized to function properly in the particular environment where
234 it is deployed. The OVF environment provides a standard and extensible way for the virtualization
235 platform to communicate deployment configuration to the guest software.

236 The OVF environment is an XML document containing deployment time customization information for the
237 guest software. Examples of information that could be provided in the XML document include:

- 238 • Operating system level configuration, such as host names, IP address, subnets, gateways, etc.
- 239 • Application-level configuration such as DNS name of active directory server, databases and
240 other external services.

241 The set of properties that are to be configured during deployment are specified in the OVF descriptor
242 using the ProductSection meta-data, and is typically entered by the user using a Wizard style interface
243 during deployment.

244 For instance, the OVF environment allows guest software to automate the network settings between
245 multi-tiered services, and the web server may automatically configure itself with the IP address of the
246 database server without any manual user interaction.

247 Defining a standard OVF environment does pose some challenges, since no standard cross-vendor para-
 248 virtualized device exists for communicating between the guest software running in a virtual machine and
 249 the underlying virtualization platform. The approach taken by the OVF specification is to split the OVF
 250 environment definitions into two parts: i) A standard *protocol* that specifies what information is available
 251 and what format it is available in, and ii) a *transport*, that specifies how the information is obtained.

252 The specification requires all implementations to support an ISO transport, which will make the OVF
 253 environment (XML document) available to the guest software on a dynamically generated ISO image.

254 See appendix A and B for examples of OVF environment documents.

255

256 2.3 Sample OVF Descriptor

257 The following listing shows a complete OVF descriptor for a typical single virtual machine appliance:

```

258 <?xml version="1.0" encoding="UTF-8"?>
259 <Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
260   xmlns="http://schemas.dmtf.org/ovf/1/envelope"
261   xmlns:ovf="http://schemas.dmtf.org/ovf/1/envelope"
262   xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
263   xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
264   schema/2/CIM_ResourceAllocationSettingData">
265
266   <!-- References to all external files -->
267   <References>
268     <File ovf:id="file1" ovf:href="vmdisk1.vmdk" ovf:size="180114671"/>
269   </References>
270   <!-- Describes meta-information for all virtual disks in the package -->
271   <DiskSection>
272     <Info>Describes the set of virtual disks</Info>
273     <Disk ovf:diskId="vmdisk1" ovf:fileRef="file1" ovf:capacity="4294967296"
274       ovf:format="http://www.vmware.com/interfaces/specifications/vmdk.html#sparse"/>
275   </DiskSection>
276   <!-- Describes all networks used in the package -->
277   <NetworkSection>
278     <Info>List of logical networks used in the package</Info>
279     <Network ovf:name="VM Network">
280       <Description>The network that the service will be available on</Description>
281     </Network>
282   </NetworkSection>
283   <VirtualSystem ovf:id="vm">
284     <Info>Describes a virtual machine</Info>
285     <Name>Virtual Appliance One</Name>
286     <ProductSection>
287       <Info>Describes product information for the appliance</Info>
288       <Product>The Great Appliance</Product>
289       <Vendor>Some Great Corporation</Vendor>
290       <Version>13.00</Version>
291       <FullVersion>13.00-b5</FullVersion>
292       <ProductUrl>http://www.somegreatcorporation.com/greatappliance</ProductUrl>
293       <VendorUrl>http://www.somegreatcorporation.com/</VendorUrl>
294       <Property ovf:key="admin.email" ovf:type="string">
295         <Description>Email address of administrator</Description>
296       </Property>
297       <Property ovf:key="app.ip" ovf:type="string" ovf:defaultValue="192.168.0.10">
298         <Description>The IP address of this appliance</Description>
299       </Property>
300     </ProductSection>
301     <AnnotationSection ovf:required="false">
302       <Info>A random annotation on this service. It can be ignored</Info>
303       <Annotation>Contact customer support if you have any problems</Annotation>
304     </AnnotationSection>
305     <EulaSection>
306       <Info>License information for the appliance</Info>
307       <License>Insert your favorite license here</License>
308     </EulaSection>
309     <VirtualHardwareSection>
310       <Info>256MB, 1 CPU, 1 disk, 1 nic</Info>
311       <Item>

```

```

312     <rasd:Description>Number of virtual CPUs</rasd:Description>
313     <rasd:ElementName>1 virtual CPU</rasd:ElementName>
314     <rasd:InstanceID>1</rasd:InstanceID>
315     <rasd:ResourceType>3</rasd:ResourceType>
316     <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
317   </Item>
318   <Item>
319     <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
320     <rasd:Description>Memory Size</rasd:Description>
321     <rasd:ElementName>256 MB of memory</rasd:ElementName>
322     <rasd:InstanceID>2</rasd:InstanceID>
323     <rasd:ResourceType>4</rasd:ResourceType>
324     <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
325   </Item>
326   <Item>
327     <rasd:AutomaticAllocation>true</rasd:AutomaticAllocation>
328     <rasd:Connection>VM Network</rasd:Connection>
329     <rasd:ElementName>Ethernet adapter on "VM Network"</rasd:ElementName>
330     <rasd:InstanceID>4000</rasd:InstanceID>
331     <rasd:ResourceType>10</rasd:ResourceType>
332   </Item>
333   <Item>
334     <rasd:ElementName>Harddisk 1</rasd:ElementName>
335     <rasd:HostResource>ovf:/disk/vmdisk1</rasd:HostResource>
336     <rasd:InstanceID>22001</rasd:InstanceID>
337     <rasd:ResourceType>17</rasd:ResourceType>
338   </Item>
339 </VirtualHardwareSection>
340 <OperatingSystemSection ovf:id="58" ovf:required="false">
341   <Info>Guest Operating System</Info>
342   <Description>Windows 2000 Advanced Server</Description>
343 </OperatingSystemSection>
344 </VirtualSystem>
345 </Envelope>

```

346 Most of the descriptor is boilerplate. It starts out by describing the set of files in addition to the descriptor
 347 itself. In this case there is a single file (vmdisk1.vmdk). It then describes the set of virtual disks and the
 348 set of networks used by the appliance. Each file, disk, and network resource is given a unique identifier.
 349 These are all in separate namespaces, but the best practice is to use distinct names.
 350

351 The content of the example OVF is a single virtual machine. The content contains 5 sections:

- 352 • *ProductSection*, which provides product information such as name and vendor of the appliance
 353 and a set of properties that can be used to customize the appliance. These properties will be
 354 configured at installation time of the appliance, typically by prompting the user. This is discussed
 355 in more detail below.
- 356 • *AnnotationSection*, which is a free form annotation.
- 357 • *EulaSection*, the licensing terms for the appliance. This is typically shown during install.
- 358 • *HardwareSection*, which describes the virtual hardware. This is a required section that describes
 359 the kind of virtual hardware and set of devices that the virtual machine requires. In this particular
 360 case, a fairly typical set of hardware (500 MB of guest memory, 1 CPU, 1 NIC, and one virtual
 361 disk) is specified. The network and disk identifiers from the outer sections are referenced here.
- 362 • *OperatingSystemSection*, which describes the guest operating system.
 363

364 3 Using the Open Virtualization Format

365 3.1 Creation

366 The creation of an OVF involves the i) packaging of a set of VMs onto a set of virtual disks, ii)
 367 appropriately encoding those virtual disks, iii) attaching an OVF descriptor with a specification of the

368 virtual hardware, licensing, and other customization metadata, and iv) optionally digitally signing the
369 package. The process of installing or importing an OVF occurs when a virtualization platform consumes
370 the OVF and creates a set of virtual machines from its contents.

371 Creating an OVF can be made as simple as exporting an existing virtual machine from a virtualization
372 platform into an OVF package, and adding to it the relevant meta-data needed to correctly install and
373 execute it. This will transform the virtual machine from its current runtime state on a particular hypervisor
374 into an OVF package. During this process, the virtual machine's disks may be compressed to make it
375 more convenient to distribute.

376 For commercial-grade virtual appliances, a standard build environment may be used to produce an OVF
377 package. For example, the OVF descriptor can be managed using a source control system, and the OVF
378 package can be built using a reproducible scripting environment (such as `make` files) or, through the use
379 of appliance building toolkits that are available from multiple vendors.

380 When an OVF is created, it must be accompanied with appliance-specific post-installation configuration
381 metadata. This includes metadata for optional localization of the interface language(s) of the appliance,
382 review/signoff and/or enforcement of the EULA, and resource configuration. It can also involve the
383 addition of special drivers, agents and other tools to the guest to enhance (for example) I/O, timekeeping,
384 memory management, monitoring and orderly shutdown.

385 3.2 Deployment

386 Deployment transforms the virtual machines in an OVF package into the runtime format understood by
387 the target virtualization platform, with the appropriate resource assignments and supported by the correct
388 virtual hardware. During deployment, the platform validates the OVF integrity, making sure that the OVF
389 package has not been modified in transit, and checks that it is compatible with the local virtual hardware.
390 It also assigns resources to, and configures the virtual machines for the particular environment on the
391 target virtualization platform. This includes assigning and configuring the (physical and virtual) networks
392 to which the virtual machines must be connected; assigning storage resources for the VMs, including
393 virtual hard disks as well as any transient data sets, connections to clustered or networked storage and
394 the like; configuring CPU and memory resources, and customizing application level properties. OVF does
395 not support the conversion of guest software between processor architectures or hardware platforms.
396 Deployment instantiates one or more virtual machines with a hardware profile that is compatible with the
397 requirements captured in the OVF descriptor, and a set of virtual disks with the content specified in the
398 OVF package.

399 The deployment experience of an OVF package depends on the virtualization platform on which it is
400 deployed. It could be command-line based, scripted, or a graphical deployment wizard. The typical OVF
401 deployment tool will show or prompt for the following information:

- 402 • Show information about the OVF package (from the *ProductSection*), and ask the user to accept
403 the licensing agreement, or deal with an unattended installation.
- 404 • Validate that the virtual hardware is compatible with the specification in the OVF.
- 405 • Ask the user for the storage location of the virtual machines and what physical networks the
406 logical networks in the OVF package should be connected to.
- 407 • Ask the user to enter the specific values for the properties configured in the *ProductSection*.

408 After this configuration, it is expected that the virtual machines can be successfully started to obtain
409 (using standard procedures such as DHCP) an identity that is valid on the local network. Properties are
410 used to prompt for specific IP network configuration and other values that are particular to the deployment
411 environment. Once the appliance is booted for the first time, additional configuration of software inside
412 the appliance can be done through a management interface provided by the appliance itself, such as a
413 web interface.
414

415 4 Features

416 4.1 Virtual Hardware Description

417 The hardware description shown in section 2.3 is very general. In particular, it simply specifies that a
 418 virtual disk and a network adaptor is needed. It does not specify what the specific hardware should be.
 419 For example, a SCSI or IDE disk, or an E1000 or Vlan network card should be appropriate. More
 420 specifically, it can reasonably be assumed that if the specification is generic, then the appliance will
 421 undertake discovery of the devices present, and load relevant drivers. In this case, it must be assumed
 422 that the appliance creator has developed the appliance with a broad set of drivers, and has tested the
 423 appliance on relevant virtual hardware to ensure that it works.

424 If an OVF package is installed on a platform that does not offer the same hardware devices and/or
 425 categories of devices that are required by the guest OS that is included in the appliance, non-trivial and
 426 non-obvious installation failures can occur. The risk is not that the appliance will run incorrectly – more
 427 that it will fail to install and boot, and that the user will not be able to debug the problem. With this comes
 428 the risk of increased volume in customer support calls, and general customer dissatisfaction. A more
 429 constrained and detailed virtual hardware specification can reduce the chance of incorrect execution
 430 (since the specific devices required are listed) but this will limit the number of systems upon which the
 431 appliance will correctly install.

432 It should be borne in mind that simplicity, robustness, and predictability of installation are key reasons that
 433 ISVs are moving to the virtual appliance model, and therefore appliance developers should create
 434 appliances for which the hardware specification is more rather than less generic, unless the appliance
 435 has very specific hardware needs. At the outset, the portability of the appliance is based on the guest OS
 436 used in the virtual machines.

437 Ideally, the appliance vendor will create a virtual machine that has device drivers for the virtual hardware
 438 of all of the vendor's desired target virtualization platforms. However, many virtualization platform vendors
 439 today do not distribute drivers independently to virtual appliance vendors/creators. Instead, to further
 440 simplify the management of the virtual hardware / appliance interface, the OVF model supports an explicit
 441 installation mode, in which each virtual machine is booted once right after installation, to permit
 442 localization/customization for the specific virtualization platform. This allows the virtual machine to detect
 443 the virtualization platform and install the correct set of device drivers, including any platform specific
 444 drivers that are made available to the guest when it first re-boots (via for example, floppy or CD drives
 445 attached to the guest on first boot). In addition, for sysprepped Windows VMs, which need only re-
 446 installation and customization with naming etc, the re-boot technique allows naming and tailoring of the
 447 image to be achieved in an automated fashion.

448 Example where multiple virtual hardware profiles are specified in the same descriptor:

```
449
450 <VirtualHardwareSection>
451   <Info>500Mb, 1 CPU, 1 disk, 1 nic virtual machine</Info>
452   <System>
453     ...
454   </System>
455   <Item>
456     ...
457   </Item>
458   ...
459 </VirtualHardwareSection>
460 <VirtualHardwareSection>
461   <Info>500Mb, 1 CPU, 1 disk, 1 nic virtual machine</Info>
462   <System>
463     ...
464   </System>
465   <Item>
466     ...
467   </Item>
468   ...
469 </VirtualHardwareSection>
```

470 This allows the vendor to tailor the hardware description to support different virtualization platforms and
 471 features. A specific virtualization platform may choose between any of the specific virtual hardware
 472 sections that it can support, with the assumption that the OVF installer will choose the latest or most
 473 capable feature set that is available on the local platform.

474 Example where specific device types are specified:

```
475 <Item>
476   <rasd:ElementName>SCSI Controller 0</rasd:ElementName>
477   <rasd:InstanceID>1000</rasd:InstanceID>
478   <rasd:ResourceSubType>LsiLogic BusLogic</rasd:ResourceSubType>
479   <rasd:ResourceType>6</rasd:ResourceType>
480 </Item>
481 <Item>
482   <rasd:ElementName>Harddisk 1</rasd:ElementName>
483   <rasd:HostResource>ovf:/disk/vmdisk1</rasd:HostResource>
484   <rasd:InstanceID>22001</rasd:InstanceID>
485   <rasd:Parent>1000</rasd:Parent>
486   <rasd:ResourceType>17</rasd:ResourceType>
487 </Item>
```

488 In the above examples, the ResourceSubType is used to specify the exact devices that are supported by
 489 the guest OS in the appliance.

491 4.2 Deployment Options

492 The author of an OVF package will have the ability to include meta-data about the intended resource
 493 requirements for a virtual appliance. This is formatted as a human-readable list of configurations, for
 494 instance:

- 495 1. Software evaluation setup
- 496 2. 10-100 person workgroup setup
- 497 3. 100-1000 person workgroup setup
- 498 4. Large enterprise workgroup setup

499 The deployer of the package will be prompted to select a configuration during deployment. In addition to
 500 exact values, ranges can also be specified. For example, the memory size can be specified as being
 501 600MB, and that the recommended range is between 500MB to 1000MB. Typically, a user will not be
 502 prompted to specify a value for a range when deploying an OVF package. The list of configurations
 503 described above is expected to be used to get to a good initial resource configuration. A range
 504 specification becomes useful when the installation later needs to be changed based on different resource
 505 needs.

506 Example list of configurations:

```
507 <DeploymentOptionSection>
508   <Configuration ovf:id="min">
509     <Label>Minimal</Label>
510     <Description>Minimal setup</Description>
511   </Configuration>
512   <Configuration ovf:id="normal" ovf:default="yes">
513     <Label>Normal</Label>
514     <Description>Standard setup</Description>
515   </Configuration>
516   ... more configurations ...
517 </DeploymentOptionSection>
```

518

519 Resource requirement example:

```
520 <ResourceAllocationSection>
521   <Info>Defines reservations for CPU and memory</Info>
522   <Item>
```

```

523     ... normal configuration ...
524   </Item>
525   <Item ovf:configuration="min">
526     ... overwrites for minimal configuration ...
527   </Item>
528 </ResourceAllocationSection>
529

```

530 VirtualHardwareSection example:

```

531 <VirtualHardwareSection>
532   <Info>...</Info>
533   <Item>
534     <rasd:AllocationUnits>hertz * 10^6</rasd:AllocationUnits>
535     <rasd:ElementName>1 CPU and 500 MHz reservation</rasd:ElementName>
536     <rasd:InstanceID>1</rasd:InstanceID>
537     <rasd:Reservation>500</rasd:Reservation>
538     <rasd:ResourceType>4</rasd:ResourceType>
539     <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
540   </Item>
541   ...
542   <Item ovf:configuration="big">
543     <rasd:ElementName>1 CPU and 800 MHz reservation</rasd:ElementName>
544     <rasd:InstanceID>0</rasd:InstanceID>
545     <rasd:Reservation>600</rasd:Reservation>
546     <rasd:ResourceType>3</rasd:ResourceType>
547   </Item>
548 </VirtualHardwareSection>
549

```

550 4.3 Deployment Customization

551 The OVF descriptor can contain a description of the software product installed in the guest, including
 552 how it can be customized through the OVF environment.

```

553 <ProductSection>
554   <Info>Describes product information for the service</Info>
555   <Product>MyService Web Portal</Product>
556   <Vendor>Some Random Organization</Vendor>
557   <Version>4.5</Version>
558   <FullVersion>4.5-b4523</FullVersion>
559   <ProductUrl>http://www.vmware.com/go/ovf</ProductUrl>
560   <VendorUrl>http://www.vmware.com/</VendorUrl>
561   <Property ovf:key="adminEmail" ovf:type="string" ovf:userConfigurable="true">
562     <Description>Email address of administrator</Description>
563   </Property>
564   <Property ovf:key="appIp" ovf:type="string" ovf:userConfigurable="true">
565     <Description>IP address of the application</Description>
566   </Property>
567 </ProductSection>
568

```

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

572 Appendix 0 contains a detailed example of customization of a complex multi-tiered application.

573

574 4.4 Internationalization

575 The OVF specification support localizable messages using the optional `ovf:msgid` attribute:

```

576 <Envelope ...>
577   ...
578   <Info ovf:msgid="info.os">Operating System</Info>
579   ...
580   <Strings xml:lang="da-DA">
581

```

```

582 <Msg ovf:msgid="info.os">Operativsystem</Msg>
583 ...
584 </Strings>
585 <Strings xml:lang="de-DE">
586 <Msg ovf:msgid="info.os">Betriebssystem</Msg>
587 ...
588 </Strings>
589 </Envelope>

```

In the example above the localized strings are stored inside the OVF descriptor, but localized strings may also be stored outside the OVF descriptor using external string bundles. For example:

```

593 <Envelope ...>
594 <References>
595 ...
596 <File ovf:id="da-DA-resources" ovf:href="german.msg" />
597 <File ovf:id="de-DE-resources" ovf:href="danish.msg" />
598 ...
599 </References>
600 ...
601 <Strings xml:lang="da-DA" ovf:fileRef="da-da-resources" />
602 <Strings xml:lang="de-DE" ovf:fileRef="de-de-resources" />
603 </Envelope>

```

4.5 Extensibility

A design goal of the OVF specification is to ensure backwards- and forwards compatibility. For forwards compatibility, this means that an OVF descriptor using features of a later specification (or custom extensions) can be understood by an OVF consumer that is written to either i) an earlier version of the specification, or ii) has no knowledge of the particular extensions. OVF consumer should be able to reliably, predictably, and in a user-friendly manner, decide whether to reject or accept an OVF package that contains extensions.

OVF supports an open-content model that allows additional sections to be added, as well as allowing existing sections to be extended with new content. On extensions, a Boolean `ovf:required` attribute specifies whether the information in the element is required for correct behavior or optional.

Example of adding new section:

```

618 <ns:BuildInformationSection ovf:required="false">
619 <Info>Specifies information on how a virtual machine was created</Info>
620 <BuildNumber> ... </BuildNumber >
621 <BuildDate> ... </BuildDate >
622 <BuildSystem> ... </BuildSystem>
623 ...
624 </ns:BuildInformationSection>

```

Example of extending existing section:

```

628 <AnnotationSection>
629 <Info>Specifies an annotation for this virtual machine</Info>
630 <Annotation>This is an example of how a future element (Author) can still be parsed by older
631 clients</Annotation>
632 <!-- AnnotationSection extended with Author element -->
633 <ns:Author ovf:required="false">John Smith</ns:Author>
634 </AnnotationSection>

```

See appendix C for detailed examples on OVF documents extensions.

4.6 Conformance

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

- 642 • OVF descriptor only contains meta-data defined in the OVF specification, i.e. no custom
643 extensions are present.
644 Conformance Level: 1.
- 645 • OVF descriptor contains meta-data with custom extensions, but all such extensions are optional.
646 Conformance Level: 2.
- 647 • OVF descriptor contains meta-data with custom extensions, and at least one such extension is
648 required.
649 Conformance Level: 3.

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

651

652 5 Portability

653 OVF is an enabling technology for enhancing portability of virtual appliances and their associated virtual
654 machines. An OVF package contains a recipe for creating virtual machines that can be interpreted
655 concisely by a virtualization platform. The packaged meta-data enables a robust and user-friendly
656 experience when installing a virtual appliance. In particular, the meta-data can be used by the
657 management infrastructure to confidently decide whether a particular VM described in an OVF can be
658 installed or whether it should be rejected, and potentially to guide appropriate conversions and
659 localizations to make it runnable in the specific execution context in which it is to be installed.

660 There are many factors that are beyond the control of the OVF format specification and even a fully
661 compliant implementation of it, that determine the portability of a packaged virtual machine. That is, the
662 act of packaging a virtual machine into an OVF package does not guarantee universal portability or
663 install-ability across all hypervisors. Below are some of the factors that could limit portability:

- 664 • The VMs in the OVF could contain virtual disks in a format that is not understood by the
665 hypervisor attempting the installation. While it is reasonable to expect that most hypervisors will
666 be able to import and/or export VMs in any of the major virtual hard disk formats, newer formats
667 may arise that are supported by the OVF and not a particular hypervisor. It may be useful in
668 future versions of this specification, to stipulate a required set of virtual hard disk formats that
669 must be supported by an OVF compliant hypervisor.
- 670 • The installed guest software may not support the virtual hardware presented by the hypervisor.
671 By way of example, the Xen hypervisor does not by default offer a virtualized floppy disk device to
672 guests. One could conceive of a guest VM that would require interaction with a floppy disk
673 controller and which therefore would not be able to execute the VM correctly.
- 674 • The installed guest software does not support the CPU architecture. For example, the guest
675 software might execute CPU operations specific to certain processor models or require specific
676 floating point support, or contain opcodes specific to a particular vendor's CPU.
- 677 • The virtualization platform might not understand a feature requested in the OVF descriptor. For
678 example, composed services may not be supported. Since the OVF standard will evolve
679 independently of virtualization products, at any point an OVF might be unsupported on a
680 virtualization platform that pre-dates that OVF specification.

681 The portability of an OVF can be categorized into the following 3 levels:

- 682 • **Level 1.** Only runs on a particular virtualization product and/or CPU architecture and/or virtual
683 hardware selection. This would typically be due to the OVF containing suspended virtual
684 machines or snapshots of powered on virtual machines, including the current run-time state of the
685 CPU and real or emulated devices. Such state ties the OVF to a very specific virtualization and
686 hardware platform.

687 • **Level 2.** Runs on a specific family of virtual hardware. This would typically be due to lack of driver
688 support by the installed guest software.

689 • **Level 3.** Runs on multiple families of virtual hardware. For example, the appliance could be
690 runnable on Xen, Sun, Microsoft, and VMware hypervisors. For level 3 compatibility, the guest
691 software has been developed to support the devices of multiple hypervisors. A clean install and
692 boot of a guest OS, during which the guest OS performs hardware device discovery and installs
693 any specialized drivers required to interact with the virtual platform, is an example of Level 3
694 portability of an OVF. The “sysprep” level of portability for Microsoft Windows® operating
695 systems is another example. Such OS instances can be re-installed, re-named and re-
696 personalized on multiple hardware platforms, including virtual hardware.

697 For use within an organization, Level 1 or Level 2 compatibility may be good enough, since the OVF
698 package is distributed within a controlled environment where specific purchasing decisions of hardware or
699 virtualization platforms can ensure consistency of the underlying feature set for the OVF. A simple export
700 of a virtual machine will typically create an OVF with Level 1 or Level 2 compatibility (tied to a specific set
701 of virtual hardware), however it is easy to extend the metaphor to support the export of Level 3
702 compatibility, for example through the use of utilities such as “sysprep” for Windows.

703 For commercial appliances independently created and distributed by ISVs, Level 3 compatibility is highly
704 desirable. Indeed, Level 3 compatibility ensures that the appliance is readily available for the broadest
705 possible customer base both for evaluation and production. Toolkits will generally be used to create
706 certified “known good” Level 3 packages of the appliance for broad distribution and installation on multiple
707 virtual platforms, or Level 2 compatibility packages if the appliance is to be consumed within the context
708 of a narrower set of virtual hardware, such as within a particular development group in an enterprise.

709 The OVF virtual hardware description is designed to support Level 1 through Level 3 portability. For Level
710 3 portability it is possible to include only very general descriptions of hardware requirements, or to specify
711 multiple alternative virtual hardware descriptions. The appliance provider is in full control of how flexible
712 or restrictive the virtual hardware specification is made. A narrow specification can be used to constrain
713 an appliance to run on only known-good virtual hardware, while limiting its portability somewhat. A broad
714 specification makes the appliance useful across as wide a set of virtual hardware as possible. This
715 ensures that customers have the best possible user experience, which is one of the main requirements
716 for the success of the virtual appliance concept.

717

718 **6 Future Versions of the OVF Specification**

719 The scope of OVF specification version 1.0 is the packaging and deployment phases of the virtual
720 appliance software life cycle. OVF 1.0 provides the core framework that allows workflow and system-level
721 meta-data to be encoded, stored, and transported.

722 In the OVF package, information can be stored that describes how the appliance is to interact with
723 external processes and systems. Examples of such functionality are appliance upgrade, cataloging, and
724 integrity and/or security checking, dependency checking, and enhanced license management. Future
725 versions of the specification may look at standardizing such metadata.

726 An OVF package can contain multi-tiered applications, including complex nested configurations, but OVF
727 currently does not support composition of existing OVF packages. Composing existing packages can be
728 attractive when software in an existing signed OVF package is to be embedded in a new context. Future
729 versions of the specification may look at supporting this.

730

731 **7 Conclusion**

732 The OVF specification offers a portable virtual appliance format that is intended for broad adoption across
733 the IT industry. The OVF specification is intended to be immediately useful, to solve an immediate

734 business need, and to facilitate the rapid adoption of a common, backwards compatible, yet rich virtual
735 machine format. OVF is complementary to existing IT management standards and frameworks, and will
736 be further developed within a standards organization. OVF promotes customer confidence through the
737 collaborative development of common standards for portability and interchange of virtual machines
738 between different vendors' virtualization platforms, and promotes best-of-breed competition through its
739 openness and extensibility.

740 The OVF specification is intended to evolve in an appropriate standards organization. The explicit
741 copyright notice attached to this document is intended to avoid arbitrary piece-wise extensions to the
742 format outside the context of a standards organization, while permitting free distribution and
743 implementation of the specification.

744 A Multi-tiered Petstore Example

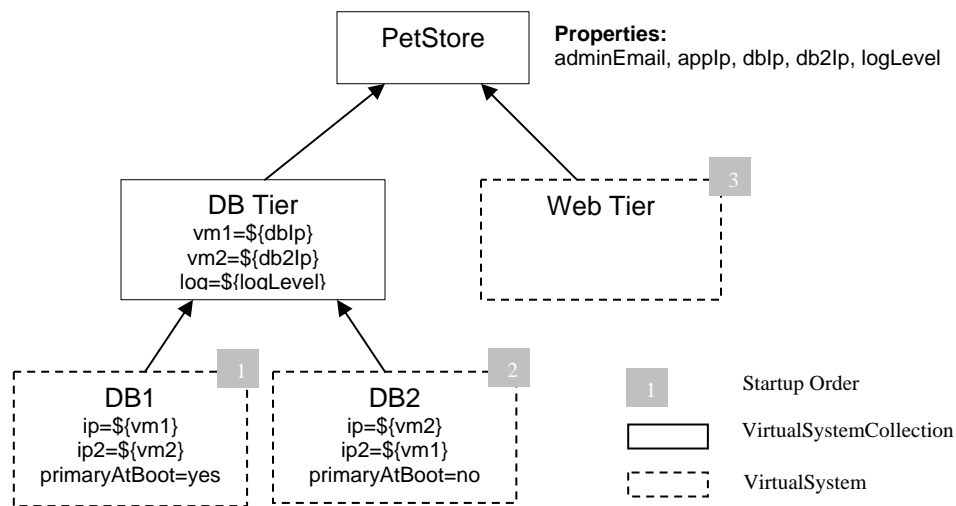
745 This example will demonstrate several advanced OVF concepts:

- 746
- 747 • Multi-VM packages - use of the VirtualMachineCollection entity subtype
- 748 • Composite service organization - use of nested VirtualMachineCollection entity subtype
- 749 • Propagation of user defined deployment configuration.
- 750 • Deployment time customization of the service using the OVF Environment.
- 751 • The use of virtual disk chains to minimize downloads.
- 752 • Nesting of ProductSections for providing information about the installed software in an individual
- 753 virtual machine

754
755 The example service is called PetStore and consists of a front-end web-server and a database. The
756 database server is itself a complex multi-tiered server consisting of two VMs for fault-tolerance.

757 Architecture and Packaging

758 The Petstore OVF package consists of 3 virtual systems (WebTier, DB1, and DB2) and 2 virtual system
759 collections (Petstore and DBTier). The diagram below shows the structure of the OVF package as well as
760 the properties and startup order of the virtual machines:
761



762 The complete OVF descriptor is listed at the end of this document. The use of properties and disk layout
763 of the OVF is discussed in more details in the following.
764
765

766 Properties

767 The Petstore service has 5 user-configurable properties. These are the key control parameters for the
768 service that needs to be configured in order for it to start up correctly in the deployed environment. The
769 properties are passed up to the guest software in the form of an OVF environment document. The guest
770 software is written to read the OVF environment on startup, extract the values of the properties, and apply
771 them to the software configuration. Thus, the OVF descriptor reflects the properties that are handled by
772 the guest software.
773

774
775 For this particular service, there are two different software configurations, one for the Web tier and one for
776 the Database tier. The properties supported in each software configuration are:

777
778 Web Guest Software:
779

Property	Description
<i>applp</i>	IP address of the WebServer.
<i>dblp</i>	IP address of the database server to connect to.
<i>adminEmail</i>	Email address for support
<i>logLevel</i>	Logging level

780
781 All properties defined on the immediate parent VirtualSystemCollection container is available to a child
782 VirtualSystem or VirtualSystemCollection. Thus, the OVF descriptor does not need to contain an explicit
783 ProductSection each VM, as demonstrated for WebVM.

784
785 Database Guest Software:
786

Property	Description
<i>ip</i>	IP address of the virtual machine
<i>primaryAtBoot</i>	Whether the instance should act as the primary or secondary when booting
<i>ip2</i>	IP address of the twin database VM that acts as the hot-spare or primary
<i>log</i>	Here the logging level is called log

787
788 The clustered database is organized as a virtual system collection itself with a specific set of properties
789 for configuration: vm1, vm2, and log. This organization separates the database implementation from the
790 rest of the software in the OVF package and allows virtual appliances (guest software + virtual machine
791 configurations) to be easily composed and thereby promotes reuse.

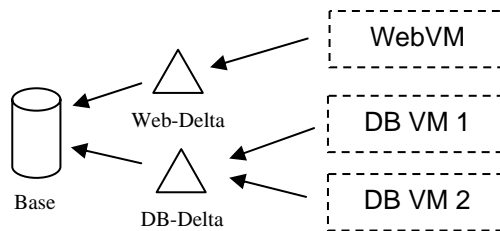
792
793 The database software is an off-the-shelf software package and the vendor has chosen the
794 "com.mydb.db" as the unique name for all the properties. This can be seen in the OVF descriptor with the
795 inclusion of the *ovf:class* attribute on the ProductSection.

796
797 The $\${<name>}$ property syntax is used to propagate values from the outer level into the inner nodes in
798 the OVF Descriptor's entity hierarchy. This mechanism allows linking up different components without
799 having to pre-negotiate naming conventions or changing guest software. Only properties defined on the
800 immediate parent VirtualSystemCollection container are available to a child entity. Thus, properties
801 defined on Petstore will not be available to a DB1. This ensures that the interface for a
802 VirtualSystemCollection is encapsulated and well described in its parent VirtualSystemCollection, which
803 makes the software composable and easy to reuse.

804
805 The OVF descriptor uses fixed non-user assignable properties to ensure that the two database virtual
806 machines boots up into different roles even though they are, initially, booting of the exact same software
807 image. The property named *com.mydb.db.primaryAtBoot* is specified with a fixed, non-user configurable
808 value but is different value for the two images. The software inspects this at boot time and customizes its
809 operation accordingly.
810

811 Disk Layout

812
813 The Petstore OVF package uses the ability to share disks and encode a delta disk hierarchy to minimize
814 the size and thereby the download time for the package. In this particular case, we only have two different
815 images (Database and Web), and if we further assume they are build on top of the same base OS
816 distribution, we can encode this in the OVF descriptor as.



817 Thus, while the package contains 3 distinct virtual machines, the total download size will be significantly
 818 smaller. In fact, only one full VM and then two relative small deltas need to be downloaded.
 819

820
 821 The physical layout of the virtual disks on the deployment system is independent of the disk structure in
 822 the OVF package. The OVF package describes the size of the virtual disk and the content (i.e., bits that
 823 needs to be on the disk). It also specifies that each virtual machine must get independent disks. Thus, a
 824 virtualization platform could install the above package as a 3 VMs with 3 independent flat disks, or it could
 825 chose to replicate the above organization, or something third, as long as each virtual machine sees a disk
 826 with the content described on initial boot and that changes written by one virtual machine does not affect
 827 the others.
 828

829 Complete OVF Descriptor

```

830 <?xml version="1.0" encoding="UTF-8"?>
831 <Envelope
832   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
833   xmlns="http://schemas.dmtf.org/ovf/envelope/1"
834   xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1"
835   xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
836   xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
837 schema/2/CIM_ResourceAllocationSettingData"
838   <!-- References to all external files -->
839   <References>
840     <File ovf:id="base" ovf:href="base.vmdk" ovf:size="180114671"/>
841     <File ovf:id="webdelta" ovf:href="webapp-delta.vmdk" ovf:size="123413"/>
842     <File ovf:id="dbdelta" ovf:href="dbapp-delta.vmdk" ovf:size="343243"/>
843   </References>
844   <!-- Describes meta-information about all virtual disks in the package.
845     This example is encoded as a delta-disk hierarchy.
846     -->
847   <DiskSection>
848     <Info>Describes the set of virtual disks</Info>
849     <Disk ovf:diskId="base" ovf:fileRef="base" ovf:capacity="4294967296"
850       ovf:populatedSize="1924967692"
851       ovf:format="http://www.vmware.com/specifications/vmdk.html#streamOptimized"/>
852     <Disk ovf:diskId="web" ovf:fileRef="webappdelta" ovf:parentRef="base"
853       ovf:capacity="4294967296"
854       ovf:format="http://www.vmware.com/specifications/vmdk.html#streamOptimized"/>
855     <Disk ovf:diskId="db" ovf:fileRef="dbdelta" ovf:parentRef="base"
856       ovf:capacity="4294967296"
857       ovf:format="http://www.vmware.com/specifications/vmdk.html#streamOptimized"/>
858   </DiskSection>
859   <!-- Describes all networks used in the package -->
860   <NetworkSection>
861     <Info>List of logical networks used in the package</Info>
862     <Network ovf:name="VM Network">
863       <Description ovf:msgid="network.description">The network that the service
864         will be available on</Description>
865     </Network>
866   </NetworkSection>
867   <!-- Deployment options for the packages -->
868   <DeploymentOptionSection>
869     <Info>List of deployment options available in the package</Info>
870     <Configuration ovf:id="minimal">
871       <Label ovf:msgid="minimal.label">Minimal</Label>
872
  
```

```

873     <Description ovf:msgid="minimal.description">Deploy service with minimal
874         resource use</Description>
875 </Configuration>
876 <Configuration ovf:id="standard" ovf:default="true">
877     <Label ovf:msgid="standard.label">Standard</Label>
878     <Description ovf:msgid="standard.description">Deploy service with standard
879         resource use</Description>
880 </Configuration>
881 </DeploymentOptionSection>
882 <!-- PetStore Virtual System Collection -->
883 <VirtualSystemCollection ovf:id="PetStore">
884     <Info>The packaging of the PetStoreService multi-tier application</Info>
885     <Name>PetStore Service</Name>
886 <!-- Overall information about the product -->
887 <ProductSection>
888     <Info>Describes product information for the service</Info>
889     <Product>PetStore Web Portal</Product>
890     <Vendor>Some Random Organization</Vendor>
891     <Version>4.5</Version>
892     <FullVersion>4.5-b4523</FullVersion>
893     <ProductUrl>http://www.vmware.com/go/ovf</ProductUrl>
894     <VendorUrl>http://www.vmware.com/</VendorUrl>
895     <Category ovf:msgid="category.email">Email properties</Category>
896     <Property ovf:key="adminEmail" ovf:type="string" ovf:userConfigurable="true">
897         <Label ovf:msgid="property.email.label">Admin email</Label>
898         <Description ovf:msgid="property.email.description">Email address of
899             service administrator</Description>
900     </Property>
901     <Category ovf:msgid="category.network">Network properties</Category>
902     <Property ovf:key="appIp" ovf:type="string"
903         ovf:userConfigurable="true">
904         <Label ovf:msgid="property.appip.label">IP</Label>
905         <Description ovf:msgid="property.appip.description">IP address of the
906             service</Description>
907     </Property>
908     <Property ovf:key="dbIp" ovf:type="string" ovf:userConfigurable="true">
909         <Label ovf:msgid="property.dpip.label">IP for DB</Label>
910         <Description ovf:msgid="property.dpip.description">Primary IP address of
911             the database</Description>
912     </Property>
913     <Property ovf:key="db2Ip" ovf:type="string"
914         ovf:userConfigurable="true">
915         <Label ovf:msgid="property.dpip2.label">IP for DB2</Label>
916         <Description ovf:msgid="property.dpip2.description">A secondary IP
917             address for the database</Description>
918     </Property>
919     <Category ovf:msgid="category.logging">Logging properties</Category>
920     <Property ovf:key="logLevel" ovf:type="string" ovf:value="normal"
921         ovf:userConfigurable="true">
922         <Label ovf:msgid="property.loglevel.label">Loglevel</Label>
923         <Description ovf:msgid="property.loglevel.description">Logging level for
924             the service</Description>
925         <Value ovf:value="low" ovf:configuration="minimal"/>
926     </Property>
927 </ProductSection>
928 <AnnotationSection ovf:required="false">
929     <Info>A annotation on this service</Info>
930     <Annotation ovf:msgid="annotation.annotation">Contact customer support for
931         any urgent issues</Annotation>
932 </AnnotationSection>
933 <ResourceAllocationSection ovf:required="false">
934     <Info>Defines minimum reservations for CPU and memory</Info>
935     <Item>
936         <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
937         <rasd:ElementName>512 MB reservation</rasd:ElementName>
938         <rasd:InstanceID>0</rasd:InstanceID>
939         <rasd:Reservation>512</rasd:Reservation>
940         <rasd:ResourceType>4</rasd:ResourceType>
941     </Item>
942     <Item ovf:configuration="minimal">
943         <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
944         <rasd:ElementName>384 MB reservation</rasd:ElementName>
945         <rasd:InstanceID>0</rasd:InstanceID>
946         <rasd:Reservation>384</rasd:Reservation>

```

```

947         <rasd:ResourceType>4</rasd:ResourceType>
948     </Item>
949     <Item>
950         <rasd:AllocationUnits>MHz</rasd:AllocationUnits>
951         <rasd:ElementName>1000 MHz reservation</rasd:ElementName>
952         <rasd:InstanceID>1</rasd:InstanceID>
953         <rasd:Reservation>500</rasd:Reservation>
954         <rasd:ResourceType>3</rasd:ResourceType>
955     </Item>
956     <Item ovf:bound="min">
957         <rasd:AllocationUnits>MHz</rasd:AllocationUnits>
958         <rasd:ElementName>500 MHz reservation</rasd:ElementName>
959         <rasd:InstanceID>1</rasd:InstanceID>
960         <rasd:Reservation>500</rasd:Reservation>
961         <rasd:ResourceType>3</rasd:ResourceType>
962     </Item>
963     <Item ovf:bound="max">
964         <rasd:AllocationUnits>MHz</rasd:AllocationUnits>
965         <rasd:ElementName>1500 MHz reservation</rasd:ElementName>
966         <rasd:InstanceID>1</rasd:InstanceID>
967         <rasd:Reservation>1500</rasd:Reservation>
968         <rasd:ResourceType>3</rasd:ResourceType>
969     </Item>
970 </ResourceAllocationSection>
971 <StartupSection>
972     <Info>Specifies how the composite service is powered-on and off</Info>
973     <Item ovf:id="DBTier" ovf:order="1" ovf:startDelay="120"
974         ovf:startAction="powerOn" ovf:waitingForGuest="true" ovf:stopDelay="120"
975         ovf:stopAction="guestShutdown"/>
976     <Item ovf:id="WebTier" ovf:order="2" ovf:startDelay="120"
977         ovf:startAction="powerOn" ovf:waitingForGuest="true" ovf:stopDelay="120"
978         ovf:stopAction="guestShutdown"/>
979 </StartupSection>
980 <VirtualSystem ovf:id="WebTier">
981     <Info>The virtual machine containing the WebServer application</Info>
982     <ProductSection>
983         <Info>Describes the product information</Info>
984         <Product>Apache Webserver</Product>
985         <Vendor>Apache Software Foundation</Vendor>
986         <Version>6.5</Version>
987         <FullVersion>6.5-b2432</FullVersion>
988     </ProductSection>
989     <OperatingSystemSection ovf:id="97">
990         <Info>Guest Operating System</Info>
991         <Description>Linux 2.4.x</Description>
992     </OperatingSystemSection>
993     <VirtualHardwareSection>
994         <Info>256 MB, 1 CPU, 1 disk, 1 nic virtual machine</Info>
995         <System>
996             <vssd:ElementName>Virtual Hardware Family</vssd:ElementName>
997             <vssd:InstanceID>0</vssd:InstanceID>
998             <vssd:VirtualSystemType>vmx-04</vssd:VirtualSystemType>
999         </System>
1000     </VirtualHardwareSection>
1001     <Item>
1002         <rasd:Description>Number of virtual CPUs</rasd:Description>
1003         <rasd:ElementName>1 virtual CPU</rasd:ElementName>
1004         <rasd:InstanceID>1</rasd:InstanceID>
1005         <rasd:ResourceType>3</rasd:ResourceType>
1006         <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
1007     </Item>
1008     <Item>
1009         <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1010         <rasd:Description>Memory Size</rasd:Description>
1011         <rasd:ElementName>256 MB of memory</rasd:ElementName>
1012         <rasd:InstanceID>2</rasd:InstanceID>
1013         <rasd:ResourceType>4</rasd:ResourceType>
1014         <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
1015     </Item>
1016     <Item>
1017         <rasd:AutomaticAllocation>true</rasd:AutomaticAllocation>
1018         <rasd:Connection>VM Network</rasd:Connection>
1019         <rasd:ElementName>Ethernet adapter on "VM Network"</rasd:ElementName>
1020         <rasd:InstanceID>3</rasd:InstanceID>
1021         <rasd:ResourceSubType>PCNet32</rasd:ResourceSubType>

```



```

1021         <rasd:ResourceType>10</rasd:ResourceType>
1022     </Item>
1023     <Item>
1024         <rasd:AddressOnParent>1</rasd:AddressOnParent>
1025         <rasd:ElementName>SCSI Controller 0 - LSI Logic</rasd:ElementName>
1026         <rasd:InstanceID>1000</rasd:InstanceID>
1027         <rasd:ResourceSubType>LsiLogic</rasd:ResourceSubType>
1028         <rasd:ResourceType>6</rasd:ResourceType>
1029     </Item>
1030     <Item>
1031         <rasd:AddressOnParent>0</rasd:AddressOnParent>
1032         <rasd:ElementName>Harddisk 1</rasd:ElementName>
1033         <rasd:HostResource>ovf:/disk/web</rasd:HostResource>
1034         <rasd:InstanceID>22001</rasd:InstanceID>
1035         <rasd:Parent>1000</rasd:Parent>
1036         <rasd:ResourceType>17</rasd:ResourceType>
1037     </Item>
1038 </VirtualHardwareSection>
1039 </VirtualSystem>
1040 <!-- Database Tier -->
1041 <VirtualSystemCollection ovf:id="DBTier">
1042     <Info>Describes a clustered database instance</Info>
1043     <ProductSection ovf:class="com.mydb.db">
1044         <Info>Product Information</Info>
1045         <Product>Somebody Clustered SQL Server</Product>
1046         <Vendor>TBD</Vendor>
1047         <Version>2.5</Version>
1048         <FullVersion>2.5-b1234</FullVersion>
1049         <Property ovf:key="vm1" ovf:value="{dbIp}" ovf:type="string"/>
1050         <Property ovf:key="vm2" ovf:value="{db2Ip}" ovf:type="string"/>
1051         <Property ovf:key="log" ovf:value="{logLevel}" ovf:type="string"/>
1052     </ProductSection>
1053     <StartupSection>
1054         <Info>Specifies how the composite service is powered-on and off</Info>
1055         <Item ovf:id="DB1" ovf:order="1" ovf:startDelay="120"
1056             ovf:startAction="powerOn" ovf:waitingForGuest="true"
1057             ovf:stopDelay="120" ovf:stopAction="guestShutdown"/>
1058         <Item ovf:id="DB2" ovf:order="2" ovf:startDelay="120"
1059             ovf:startAction="powerOn" ovf:waitingForGuest="true"
1060             ovf:stopDelay="120" ovf:stopAction="guestShutdown"/>
1061     </StartupSection>
1062     <!-- DB VM 1 -->
1063     <VirtualSystem ovf:id="DB1">
1064         <Info>Describes a virtual machine with the database image installed</Info>
1065         <Name>Database Instance I</Name>
1066         <ProductSection ovf:class="com.mydb.db">
1067             <Info>Specifies the OVF properties available in the OVF environment</Info>
1068             <Property ovf:key="ip" ovf:value="{vm1}" ovf:type="string"/>
1069             <Property ovf:key="ip2" ovf:value="{vm2}" ovf:type="string"/>
1070             <Property ovf:key="primaryAtBoot" ovf:value="yes" ovf:type="string"/>
1071         </ProductSection>
1072         <VirtualHardwareSection>
1073             <Info>256 MB, 1 CPU, 1 disk, 1 nic virtual machine</Info>
1074             <System>
1075                 <vssd:ElementName>Virtual Hardware Family</vssd:ElementName>
1076                 <vssd:InstanceID>0</vssd:InstanceID>
1077                 <vssd:VirtualSystemType>vmx-04</vssd:VirtualSystemType>
1078             </System>
1079             <Item>
1080                 <rasd:Description>Number of virtual CPUs</rasd:Description>
1081                 <rasd:ElementName>1 virtual CPU</rasd:ElementName>
1082                 <rasd:InstanceID>1</rasd:InstanceID>
1083                 <rasd:ResourceType>3</rasd:ResourceType>
1084                 <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
1085             </Item>
1086             <Item>
1087                 <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1088                 <rasd:Description>Memory Size</rasd:Description>
1089                 <rasd:ElementName>256 MB of memory</rasd:ElementName>
1090                 <rasd:InstanceID>2</rasd:InstanceID>
1091                 <rasd:ResourceType>4</rasd:ResourceType>
1092                 <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
1093             </Item>
1094         </VirtualSystem>

```

```

1095         <rasd:AutomaticAllocation>true</rasd:AutomaticAllocation>
1096         <rasd:Connection>VM Network</rasd:Connection>
1097         <rasd:ElementName>Ethernet adapter on "VM Network"</rasd:ElementName>
1098         <rasd:InstanceID>3</rasd:InstanceID>
1099         <rasd:ResourceSubType>PCNet32</rasd:ResourceSubType>
1100         <rasd:ResourceType>10</rasd:ResourceType>
1101     </Item>
1102     <Item>
1103         <rasd:AddressOnParent>1</rasd:AddressOnParent>
1104         <rasd:ElementName>SCSI Controller 0 - LSI Logic</rasd:ElementName>
1105         <rasd:InstanceID>1000</rasd:InstanceID>
1106         <rasd:ResourceSubType>LsiLogic</rasd:ResourceSubType>
1107         <rasd:ResourceType>6</rasd:ResourceType>
1108     </Item>
1109     <Item>
1110         <rasd:AddressOnParent>0</rasd:AddressOnParent>
1111         <rasd:ElementName>Harddisk 1</rasd:ElementName>
1112         <rasd:HostResource>ovf:/disk/db</rasd:HostResource>
1113         <rasd:InstanceID>22001</rasd:InstanceID>
1114         <rasd:Parent>1000</rasd:Parent>
1115         <rasd:ResourceType>17</rasd:ResourceType>
1116     </Item>
1117 </VirtualHardwareSection>
1118 <OperatingSystemSection ovf:id="97">
1119     <Info>Guest Operating System</Info>
1120     <Description>Linux 2.4.x</Description>
1121 </OperatingSystemSection>
1122 </VirtualSystem>
1123 <!-- DB VM 2 -->
1124 <VirtualSystem ovf:id="DB2">
1125     <Info>Describes a virtual machine with the database image installed</Info>
1126     <Name>Database Instance II</Name>
1127     <ProductSection ovf:class="com.mydb.db">
1128         <Info>Specifies the OVF properties available in the OVF environment</Info>
1129         <Property ovf:key="ip" ovf:value="{vm2}" ovf:type="string"/>
1130         <Property ovf:key="ip2" ovf:value="{vml}" ovf:type="string"/>
1131         <Property ovf:key="primaryAtBoot" ovf:value="no" ovf:type="string"/>
1132     </ProductSection>
1133     <VirtualHardwareSection>
1134         <Info>256 MB, 1 CPU, 1 disk, 1 nic virtual machine</Info>
1135         <System>
1136             <vssd:ElementName>Virtual Hardware Family</vssd:ElementName>
1137             <vssd:InstanceID>0</vssd:InstanceID>
1138             <vssd:VirtualSystemType>vmx-04</vssd:VirtualSystemType>
1139         </System>
1140         <Item>
1141             <rasd:Description>Number of virtual CPUs</rasd:Description>
1142             <rasd:ElementName>1 virtual CPU</rasd:ElementName>
1143             <rasd:InstanceID>1</rasd:InstanceID>
1144             <rasd:ResourceType>3</rasd:ResourceType>
1145             <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
1146         </Item>
1147         <Item>
1148             <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1149             <rasd:Description>Memory Size</rasd:Description>
1150             <rasd:ElementName>256 MB of memory</rasd:ElementName>
1151             <rasd:InstanceID>2</rasd:InstanceID>
1152             <rasd:ResourceType>4</rasd:ResourceType>
1153             <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
1154         </Item>
1155         <Item>
1156             <rasd:AutomaticAllocation>true</rasd:AutomaticAllocation>
1157             <rasd:Connection>VM Network</rasd:Connection>
1158             <rasd:ElementName>Ethernet adapter on "VM Network"</rasd:ElementName>
1159             <rasd:InstanceID>3</rasd:InstanceID>
1160             <rasd:ResourceSubType>PCNet32</rasd:ResourceSubType>
1161             <rasd:ResourceType>10</rasd:ResourceType>
1162         </Item>
1163         <Item>
1164             <rasd:AddressOnParent>1</rasd:AddressOnParent>
1165             <rasd:ElementName>SCSI Controller 0 - LSI Logic</rasd:ElementName>
1166             <rasd:InstanceID>1000</rasd:InstanceID>
1167             <rasd:ResourceSubType>LsiLogic</rasd:ResourceSubType>
1168             <rasd:ResourceType>6</rasd:ResourceType>

```

```

1169         </Item>
1170         <Item>
1171             <rasd:AddressOnParent>0</rasd:AddressOnParent>
1172             <rasd:ElementName>Harddisk 1</rasd:ElementName>
1173             <rasd:HostResource>ovf:/disk/db</rasd:HostResource>
1174             <rasd:InstanceID>22001</rasd:InstanceID>
1175             <rasd:Parent>1000</rasd:Parent>
1176             <rasd:ResourceType>17</rasd:ResourceType>
1177         </Item>
1178     </VirtualHardwareSection>
1179     <OperatingSystemSection ovf:id="97">
1180         <Info>Guest Operating System</Info>
1181         <Description>Linux 2.4.x</Description>
1182     </OperatingSystemSection>
1183 </VirtualSystem>
1184 </VirtualSystemCollection>
1185 </VirtualSystemCollection>
1186 <!-- External I18N bundles -->
1187 <Strings xml:lang="de-DE" ovf:fileRef="de-DE-bundle.xml"/>
1188 <!-- Embedded I18N bundles -->
1189 <Strings xml:lang="da-DA">
1190     <Msg ovf:msgid="network.description">Netværket servicen skal være tilgængelig på</Msg>
1191     <Msg ovf:msgid="annotation.annotation">Kontakt kundeservice i tilfælde af
1192         kritiske problemer</Msg>
1193     <Msg ovf:msgid="property.email.description">Email adresse for administrator</Msg>
1194     <Msg ovf:msgid="property.appip.description">IP adresse for service</Msg>
1195     <Msg ovf:msgid="property.dpip">Primær IP adresse for database</Msg>
1196     <Msg ovf:msgid="property.dpip2.description">Sekundær IP adresse for database</Msg>
1197     <Msg ovf:msgid="property.loglevel.description">Logningsniveau for service</Msg>
1198     <Msg ovf:msgid="minimal.label">Minimal</Msg>
1199     <Msg ovf:msgid="minimal.description">Installer service med minimal brug af
1200         resourcer</Msg>
1201     <Msg ovf:msgid="standard.label">Normal</Msg>
1202     <Msg ovf:msgid="standard.description">Installer service med normal brug af
1203         resourcer</Msg>
1204 </Strings>
1205 </Envelope>
1206

```

1207 Complete OVF Environments

1208
1209 The following lists the OVF environments seen by the WebTier and DB1 virtual machines (DB2 is
1210 virtually identical to the one for DB1 and is omitted).

1211
1212 OVF environment for the WebTier virtual machine:

```

1213 <?xml version="1.0" encoding="UTF-8"?>
1214 <Environment
1215     xmlns="http://schemas.dmtf.org/ovf/environment/1"
1216     xmlns:ovfenv="http://schemas.dmtf.org/ovf/environment/1"
1217     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1218     ovfenv:id="WebTier">
1219
1220     <!-- Information about hypervisor platform -->
1221     <PlatformSection>
1222         <Kind>ESX Server</Kind>
1223         <Version>3.0.1</Version>
1224         <Vendor>VMware, Inc.</Vendor>
1225         <Locale>en_US</Locale>
1226     </PlatformSection>
1227
1228     <!-- Properties defined for this virtual machine -->
1229     <PropertySection>
1230         <Property ovfenv:key="adminEmail" ovfenv:value="ovf-admin@vmware.com"/>
1231         <Property ovfenv:key="appIp" ovfenv:value="10.20.132.101"/>
1232         <Property ovfenv:key="dbIp" ovfenv:value="10.20.132.102"/>
1233         <Property ovfenv:key="db2Ip" ovfenv:value="10.20.132.103"/>
1234         <Property ovfenv:key="logLevel" ovfenv:value="warning"/>
1235     </PropertySection>
1236
1237

```

```

1238 <Entity ovfenv:id="DBTier">
1239   <PropertySection>
1240     <Property ovfenv:key="adminEmail" ovfenv:value="ovf-admin@vmware.com"/>
1241     <Property ovfenv:key="appIp" ovfenv:value="10.20.132.101"/>
1242     <Property ovfenv:key="dbIp" ovfenv:value="10.20.132.102"/>
1243     <Property ovfenv:key="db2Ip" ovfenv:value="10.20.132.103"/>
1244     <Property ovfenv:key="logLevel" ovfenv:value="warning"/>
1245     <Property ovfenv:key="com.mydb.db.vml" ovfenv:value="10.20.132.102"/>
1246     <Property ovfenv:key="com.mydb.db.vm2" ovfenv:value="10.20.132.103"/>
1247     <Property ovfenv:key="com.mydb.db.log" ovfenv:value="warning"/>
1248   </PropertySection>
1249 </Entity>
1250 </Environment>

```

1251
1252 OVF environment for the DB1 virtual machine:

```

1253 <?xml version="1.0" encoding="UTF-8"?>
1254 <Environment
1255   xmlns="http://schemas.dmtf.org/ovf/environment/1"
1256   xmlns:ovfenv="http://schemas.dmtf.org/ovf/environment/1"
1257   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1258   ovfenv:id="DB1">
1259
1260   <!-- Information about hypervisor platform -->
1261   <PlatformSection>
1262     <Kind>ESX Server</Kind>
1263     <Version>3.0.1</Version>
1264     <Vendor>VMware, Inc.</Vendor>
1265     <Locale>en_US</Locale>
1266   </PlatformSection>
1267
1268   <!-- Properties defined for this virtual machine -->
1269   <PropertySection>
1270     <Property ovfenv:key="com.mydb.db.vml" ovfenv:value="10.20.132.102"/>
1271     <Property ovfenv:key="com.mydb.db.vm2" ovfenv:value="10.20.132.103"/>
1272     <Property ovfenv:key="com.mydb.db.log" ovfenv:value="warning"/>
1273     <Property ovfenv:key="com.mydb.db.ip" ovfenv:value="10.20.132.102"/>
1274     <Property ovfenv:key="com.mydb.db.ip2" ovfenv:value="10.20.132.103"/>
1275     <Property ovfenv:key="com.mydb.db.primaryAtBoot" ovfenv:value="yes"/>
1276   </PropertySection>
1277
1278   <Entity ovfenv:id="DB2">
1279     <PropertySection>
1280       <Property ovfenv:key="com.mydb.db.vml" ovfenv:value="10.20.132.102"/>
1281       <Property ovfenv:key="com.mydb.db.vm2" ovfenv:value="10.20.132.103"/>
1282       <Property ovfenv:key="com.mydb.db.log" ovfenv:value="warning"/>
1283       <Property ovfenv:key="com.mydb.db.ip" ovfenv:value="10.20.132.103"/>
1284       <Property ovfenv:key="com.mydb.db.ip2" ovfenv:value="10.20.132.102"/>
1285       <Property ovfenv:key="com.mydb.db.primaryAtBoot" ovfenv:value="no"/>
1286     </PropertySection>
1287   </Entity>
1288 </Environment>
1289
1290

```

1291

1292 B LAMP Stack Example

1293 In this example we provide two concrete examples on how an OVF descriptor for a LAMP virtual
 1294 appliance could look like. We show both a single-VM LAMP virtual appliance and a multi-VM LAMP virtual
 1295 appliance. LAMP is an abbreviation for a service built using the Linux operating system, Apache web
 1296 server, MySQL database, and the PHP web development software packages.

1297
 1298 This examples show how the *ProductSection* can be used to specify both operating system and
 1299 application-level deployment parameters. For example, these parameters can be used to optimize the
 1300 performance of a service when deployed into a particular environment. The descriptors are complete, but
 1301 otherwise kept minimal, so there are, for example, no EULA sections.
 1302

1303 Deployment-time Customization

1304 A part of the deployment phase of an OVF package is to provide customization parameters. The
 1305 customization parameters are specified in the OVF descriptor and are provided to the guest software
 1306 using the OVF environment. This deployment time customization is in addition to the virtual machine level
 1307 parameters, which includes virtual switch connectivity and physical storage location.
 1308

1309 For a LAMP-based virtual appliance, the deployment time customization includes IP address and port
 1310 number of the service, network information such as gateway and subnet, and also parameters so the
 1311 performance can be optimized for a given deployment. The properties that will be exposed to the
 1312 deployer will vary from vendor to vendor and service to service. In our example descriptors, we use the
 1313 following set of parameters for the 4 different LAMP components:
 1314

Product	Property	Description
Linux	<i>hostname</i>	Network identity of the application, including IP address.
	<i>ip</i>	
	<i>subnet</i>	
	<i>gateway</i>	
	<i>dns</i>	
	netCoreRmemMax	Parameters to optimize the transfer rate of the IP stack
netCoreWmemMax		
Apache	httpPort	Port numbers for web server
	httpsPort	
	startThreads	Parameters to optimize the performance of the web server
	minSpareThreads	
	maxSpareThreads	
maxClients		
MySQL	queryCacheSize	Parameters to optimize the performance of database
	maxConnections	
	waitTimeout	
PHP	sessionTimeout	Parameters to customize the behavior of the PHP engine, including how sessions timeout and number of sessions.
	concurrentSessions	
	memoryLimit	

1315
 1316 The parameters in *italic* are required configuration from the user. Otherwise, they have reasonable
 1317 defaults, so the user does not necessarily need to provide a value.
 1318

1319 The customization parameters for each software product are encapsulated in separate product sections.
 1320 For example, for the Apache web server the following section is used:

```
1321 <ProductSection ovf:class="org.apache.httpd">
1322   <Info>Product customization for the installed Apache Web Server</Info>
1323   <Product>Apache Distribution Y</Product>
1324
```

```

1325 <Version>2.6.6</Version>
1326 <Property ovf:key="httpPort" ovf:type="uint16" ovf:value="80"
1327     ovf:userConfigurable="true">
1328     <Description>Port number for HTTP requests</Description>
1329 </Property>
1330 <Property ovf:key="httpsPort" ovf:type="uint16" ovf:value="443"
1331     ovf:userConfigurable="true">
1332     <Description>Port number for HTTPS requests</Description>
1333 </Property>
1334 <Property ovf:key="startThreads" ovf:type="uint16" ovf:value="50"
1335     ovf:userConfigurable="true">
1336     <Description>Number of threads created on startup. </Description>
1337 </Property>
1338 <Property ovf:key="minSpareThreads" ovf:type="uint16" ovf:value="15"
1339     ovf:userConfigurable="true">
1340     <Description>Minimum number of idle threads to handle request spikes.</Description>
1341 </Property>
1342 <Property ovf:key="maxSpareThreads" ovf:type="uint16" ovf:value="30"
1343     ovf:userConfigurable="true">
1344     <Description>Maximum number of idle threads </Description>
1345 </Property>
1346 <Property ovf:key="maxClients" ovf:type="uint16" ovf:value="256"
1347     ovf:userConfigurable="true">
1348     <Description>Limit the number of simultaneous requests that will be served.
1349 </Description>
1350 </Property>
1351 </ProductSection>

```

1352 The `ovf:class="org.apache.httpd"` attribute specifies the prefix for the properties. Hence, the
 1353 Apache database is expected to look for the following properties in the OVF environment:

```

1354 <Environment
1355     ...
1356     <!-- Properties defined for this virtual machine -->
1357     <PropertySection>
1358     <Property ovfenv:name="org.apache.httpd.httpPort" ovfenv:value="80"/>
1359     <Property ovfenv:name="org.apache.httpd.httpsPort" ovfenv:value="443"/>
1360     <Property ovfenv:name="org.apache.httpd.startThreads" ovfenv:value="50"/>
1361     <Property ovfenv:name="org.apache.httpd.minSpareThreads" ovfenv:value="15"/>
1362     <Property ovfenv:name="org.apache.httpd.maxSpareThreads" ovfenv:value="30"/>
1363     <Property ovfenv:name="org.apache.httpd.maxClients" ovfenv:value="256"/>
1364     ...
1365     </PropertySection>
1366     ...
1367 </Environment>

```

1368

1369 Simple LAMP OVF Descriptor

1370 A complete OVF descriptor for a single VM virtual appliance with the LAMP stack is listed below:

```

1371 <?xml version="1.0" encoding="UTF-8"?>
1372 <Envelope
1373     xmlns="http://schemas.dmtf.org/ovf/envelope/1"
1374     xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1"
1375     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1376     xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
1377     xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
1378 schema/2/CIM_ResourceAllocationSettingData"
1379     <!-- References to all external files -->
1380     <References>
1381     <File ovf:id="lamp" ovf:href="lamp.vmdk" ovf:size="180114671"/>
1382     </References>
1383     <!-- Describes meta-information about all virtual disks in the package. -->
1384     <DiskSection>
1385     <Info>List of the virtual disks used in the package</Info>
1386     <Disk ovf:diskId="lamp" ovf:fileRef="lamp" ovf:capacity="4294967296"
1387     ovf:populatedSize="1924967692"
1388     ovf:format="http://www.vmware.com/specifications/vmdk.html#streamOptimized"/>
1389     </DiskSection>
1390     <!-- Describes all networks used in the package -->
1391     <NetworkSection>

```

```

1392     <Info>Logical networks used in the package</Info>
1393     <Network ovf:name="VM Network">
1394         <Description>The network that the LAMP Service will be available
1395         on</Description>
1396     </Network>
1397 </NetworkSection>
1398 <VirtualSystem ovf:id="MyLampService">
1399     <Info>Single-VM Virtual appliance with LAMP stack</Info>
1400     <Name>LAMP Virtual Appliance</Name>
1401     <!-- Overall information about the product -->
1402     <ProductSection>
1403         <Info>Product information for the service</Info>
1404         <Product>Lamp Service</Product>
1405         <Version>1.0</Version>
1406         <FullVersion>1.0.0</FullVersion>
1407     </ProductSection>
1408     <!-- Linux component configuration parameters -->
1409     <ProductSection ovf:class="org.linuxdistx">
1410         <Info>Product customization for the installed Linux system</Info>
1411         <Product>Linux Distribution X</Product>
1412         <Version>2.6.3</Version>
1413         <Property ovf:key="hostname" ovf:type="string">
1414             <Description>Specifies the hostname for the appliance</Description>
1415         </Property>
1416         <Property ovf:key="ip" ovf:type="string">
1417             <Description>Specifies the IP address for the appliance</Description>
1418         </Property>
1419         <Property ovf:key="subnet" ovf:type="string">
1420             <Description> Specifies the subnet to use on the deployed network
1421             </Description>
1422         </Property>
1423         <Property ovf:key="gateway" ovf:type="string">
1424             <Description> Specifies the gateway on the deployed network
1425             </Description>
1426         </Property>
1427         <Property ovf:key="dns" ovf:type="string">
1428             <Description> A comma separated list of DNS servers on the deployed
1429             network </Description>
1430         </Property>
1431         <Property ovf:key="netCoreRmemMaxMB" ovf:type="uint16" ovf:value="16"
1432             ovf:userConfigurable="true">
1433             <Description> Specify TCP read max buffer size in mega bytes. Default is
1434             16. </Description>
1435         </Property>
1436         <Property ovf:key="netCoreWmemMaxMB" ovf:type="uint16" ovf:value="16"
1437             ovf:userConfigurable="true">
1438             <Description> Specify TCP write max buffer size in mega bytes. Default is
1439             16. </Description>
1440         </Property>
1441     </ProductSection>
1442     <!-- Apache component configuration parameters -->
1443     <ProductSection ovf:class="org.apache.httpd">
1444         <Info>Product customization for the installed Apache Web Server</Info>
1445         <Product>Apache Distribution Y</Product>
1446         <Version>2.6.6</Version>
1447         <Property ovf:key="httpPort" ovf:type="uint16" ovf:value="80"
1448             ovf:userConfigurable="true">
1449             <Description>Port number for HTTP requests</Description>
1450         </Property>
1451         <Property ovf:key="httpsPort" ovf:type="uint16" ovf:value="443"
1452             ovf:userConfigurable="true">
1453             <Description>Port number for HTTPS requests</Description>
1454         </Property>
1455         <Property ovf:key="startThreads" ovf:type="uint16" ovf:value="50"
1456             ovf:userConfigurable="true">
1457             <Description>Number of threads created on startup. </Description>
1458         </Property>
1459         <Property ovf:key="minSpareThreads" ovf:type="uint16" ovf:value="15"
1460             ovf:userConfigurable="true">
1461             <Description> Minimum number of idle threads to handle request spikes.
1462             </Description>
1463         </Property>
1464         <Property ovf:key="maxSpareThreads" ovf:type="uint16" ovf:value="30"
1465             ovf:userConfigurable="true">

```

```

1466     <Description>Maximum number of idle threads </Description>
1467   </Property>
1468   <Property ovf:key="maxClients" ovf:type="uint16" ovf:value="256"
1469     ovf:userConfigurable="true">
1470     <Description>Limit the number of simultaneous requests that will be
1471       served. </Description>
1472   </Property>
1473 </ProductSection>
1474 <!-- MySQL component configuration parameters -->
1475 <ProductSection ovf:class="org.mysql.db">
1476   <Info>Product customization for the installed MySql Database Server</Info>
1477   <Product>MySQL Distribution Z</Product>
1478   <Version>5.0</Version>
1479   <Property ovf:key="queryCacheSizeMB" ovf:type="uint16" ovf:value="32"
1480     ovf:userConfigurable="true">
1481     <Description>Buffer to cache repeated queries for faster access (in
1482       MB)</Description>
1483   </Property>
1484   <Property ovf:key="maxConnections" ovf:type="uint16" ovf:value="500"
1485     ovf:userConfigurable="true">
1486     <Description>The number of concurrent connections that can be
1487       served</Description>
1488   </Property>
1489   <Property ovf:key="waitTimeout" ovf:type="uint16" ovf:value="100"
1490     ovf:userConfigurable="true">
1491     <Description>Number of seconds to wait before timing out a connection
1492       </Description>
1493   </Property>
1494 </ProductSection>
1495 <!-- PHP component configuration parameters -->
1496 <ProductSection ovf:class="net.php">
1497   <Info>Product customization for the installed PHP component</Info>
1498   <Product>PHP Distribution U</Product>
1499   <Version>5.0</Version>
1500   <Property ovf:key="sessionTimeout" ovf:type="uint16" ovf:value="5"
1501     ovf:userConfigurable="true">
1502     <Description> How many minutes a session has to be idle before it is
1503       timed out </Description>
1504   </Property>
1505   <Property ovf:key="concurrentSessions" ovf:type="uint16" ovf:value="500"
1506     ovf:userConfigurable="true">
1507     <Description> The number of concurrent sessions that can be served
1508       </Description>
1509   </Property>
1510   <Property ovf:key="memoryLimit" ovf:type="uint16" ovf:value="32"
1511     ovf:userConfigurable="true">
1512     <Description> How much memory in megabytes a script can consume before
1513       being killed </Description>
1514   </Property>
1515 </ProductSection>
1516 <OperatingSystemSection ovf:id="99">
1517   <Info>Guest Operating System</Info>
1518   <Description>Linux 2.6.x</Description>
1519 </OperatingSystemSection>
1520 <VirtualHardwareSection>
1521   <Info>Virtual Hardware Requirements: 256MB, 1 CPU, 1 disk, 1 NIC</Info>
1522   <System>
1523     <vssd:ElementName>Virtual Hardware Family</vssd:ElementName>
1524     <vssd:InstanceID>0</vssd:InstanceID>
1525     <vssd:VirtualSystemType>vmx-04</vssd:VirtualSystemType>
1526   </System>
1527   <Item>
1528     <rasd:Description>Number of virtual CPUs</rasd:Description>
1529     <rasd:ElementName>1 virtual CPU</rasd:ElementName>
1530     <rasd:InstanceID>1</rasd:InstanceID>
1531     <rasd:ResourceType>3</rasd:ResourceType>
1532     <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
1533   </Item>
1534   <Item>
1535     <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1536     <rasd:Description>Memory Size</rasd:Description>
1537     <rasd:ElementName>256 MB of memory</rasd:ElementName>
1538     <rasd:InstanceID>2</rasd:InstanceID>
1539     <rasd:ResourceType>4</rasd:ResourceType>

```



```

1540         <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
1541     </Item>
1542     <Item>
1543         <rasd:AutomaticAllocation>true</rasd:AutomaticAllocation>
1544         <rasd:Connection>VM Network</rasd:Connection>
1545         <rasd:ElementName>Ethernet adapter on "VM Network"</rasd:ElementName>
1546         <rasd:InstanceID>3</rasd:InstanceID>
1547         <rasd:ResourceType>10</rasd:ResourceType>
1548     </Item>
1549     <Item>
1550         <rasd:ElementName>SCSI Controller 0 - LSI Logic</rasd:ElementName>
1551         <rasd:InstanceID>4</rasd:InstanceID>
1552         <rasd:ResourceSubType>LsiLogic</rasd:ResourceSubType>
1553         <rasd:ResourceType>6</rasd:ResourceType>
1554     </Item>
1555     <Item>
1556         <rasd:ElementName>Harddisk 1</rasd:ElementName>
1557         <rasd:HostResource>ovf:/disk/lamp</rasd:HostResource>
1558         <rasd:InstanceID>5</rasd:InstanceID>
1559         <rasd:Parent>4</rasd:Parent>
1560         <rasd:ResourceType>17</rasd:ResourceType>
1561     </Item>
1562 </VirtualHardwareSection>
1563 </VirtualSystem>
1564 </Envelope>

```

1565

1566 Two-tier LAMP OVF Descriptor

1567 In a two tier LAMP stack, the application tier (Linux, Apache, PHP) and the database tier (Linux, MySQL)
 1568 server) are run as separate virtual machines for greater scalability.

1569 The OVF format makes it largely transparent to the user how a service is implemented. In particular, the
 1570 deployment experience when installing a single-VM or a two-tier LAMP appliance is very similar. The only
 1571 visible difference is that the user will need to supply two IP addresses and two DNS host names.

1572 As compared to the single-VM descriptor, the following changes are made:

- 1573 • All the user-configurable parameters must be put in the *VirtualSystemCollection* entity. The
 1574 ProductSections for Apache, MySQL, and PHP are unchanged from the single VM case.
- 1575 • The Linux software in the two virtual machines needs to be configured slightly different (IP and
 1576 hostname) while sharing most parameters. A new ProductSection is added to the
 1577 *VirtualSystemCollection* to prompt the user, and the $\${property}$ expression is used to assign
 1578 the values in each *VirtualSystem* entity.
- 1579 • Disk chains are used to keep the download size comparable to that of a single VM appliance.
 1580 Since the Linux installation is stored on a shared base disk, effectively only one copy of Linux
 1581 needs to be downloaded.

1582 The complete OVF descriptor is shown below:

```

1583 <?xml version="1.0" encoding="UTF-8"?>
1584 <Envelope
1585     xmlns="http://schemas.dmtf.org/ovf/envelope/1"
1586     xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1"
1587     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1588     xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
1589     xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
1590 schema/2/CIM_ResourceAllocationSettingData"
1591     <!-- References to all external files. -->
1592     <References>
1593         <File ovf:id="lamp-base" ovf:href="lampdb.vmdk" ovf:size="180114671"/>
1594         <File ovf:id="lamp-db" ovf:href="lampdb.vmdk" ovf:size="1801146"/>
1595         <File ovf:id="lamp-app" ovf:href="lampapp.vmdk" ovf:size="34311371"/>
1596     </References>
1597     <!-- Describes meta-information about all virtual disks in the package.

```

```

1598     This example is encoded as a delta-disk hierarchy.
1599 -->
1600 <DiskSection>
1601   <Info>List of the virtual disks used in the package</Info>
1602   <Disk ovf:diskId="lamp-base" ovf:fileRef="lamp-base" ovf:capacity="4294967296"
1603     ovf:populatedSize="1924967692"
1604     ovf:format="http://www.vmware.com/specifications/vmdk.html#streamOptimized"/>
1605   <Disk ovf:diskId="lamp-db" ovf:fileRef="lamp-db" ovf:capacity="4294967296"
1606     ovf:populatedSize="19249672"
1607     ovf:format="http://www.vmware.com/specifications/vmdk.html#streamOptimized"
1608     ovf:parentRef="lamp-base"/>
1609   <Disk ovf:diskId="lamp-app" ovf:fileRef="lamp-app" ovf:capacity="4294967296"
1610     ovf:populatedSize="2349692"
1611     ovf:format="http://www.vmware.com/specifications/vmdk.html#streamOptimized"
1612     ovf:parentRef="lamp-base"/>
1613 </DiskSection>
1614 <!-- Describes all networks used in the package -->
1615 <NetworkSection>
1616   <Info>Logical networks used in the package</Info>
1617   <Network ovf:name="VM Network">
1618     <Description>The network that the LAMP Service will be available
1619     on</Description>
1620   </Network>
1621 </NetworkSection>
1622 <VirtualSystemCollection ovf:id="LampService">
1623   <Info>Virtual appliance with a 2-tier distributed LAMP stack</Info>
1624   <Name>LAMP Service</Name>
1625   <!-- Overall information about the product -->
1626   <ProductSection>
1627     <Info>Product information for the service</Info>
1628     <Product>My Lamp Service</Product>
1629     <Version>1.0</Version>
1630     <FullVersion>1.0.0</FullVersion>
1631   </ProductSection>
1632   <ProductSection>
1633     <Info>Product customization for Operating System Level</Info>
1634     <Product>Linux Distribution X</Product>
1635     <Version>2.6.3</Version>
1636     <Property ovf:key="dbHostname" ovf:type="string">
1637       <Description>Specifies the hostname for database virtual
1638       machine</Description>
1639     </Property>
1640     <Property ovf:key="appHostname" ovf:type="string">
1641       <Description>Specifies the hostname for application server virtual
1642       machine</Description>
1643     </Property>
1644     <Property ovf:key="dbIp" ovf:type="string">
1645       <Description>Specifies the IP address for the database virtual
1646       machine</Description>
1647     </Property>
1648     <Property ovf:key="appIp" ovf:type="string">
1649       <Description>Specifies the IP address for application server
1650       VM</Description>
1651     </Property>
1652     <Property ovf:key="subnet" ovf:type="string">
1653       <Description> Specifies the subnet to use on the deployed network
1654       </Description>
1655     </Property>
1656     <Property ovf:key="gateway" ovf:type="string">
1657       <Description> Specifies the gateway on the deployed network
1658       </Description>
1659     </Property>
1660     <Property ovf:key="dns" ovf:type="string">
1661       <Description> A comma separated list of DNS servers on the deployed
1662       network </Description>
1663     </Property>
1664     <Property ovf:key="netCoreRmemMaxMB" ovf:type="uint16" ovf:value="16"
1665       ovf:userConfigurable="true">
1666       <Description> Specify TCP read max buffer size in mega bytes. Default is
1667       16. </Description>
1668     </Property>
1669     <Property ovf:key="netCoreWmemMaxMB" ovf:type="uint16" ovf:value="16"
1670       ovf:userConfigurable="true">
1671       <Description> Specify TCP write max buffer size in mega bytes. Default is

```

```

1672         16. </Description>
1673     </Property>
1674 </ProductSection>
1675 <!-- Apache component configuration parameters -->
1676 <ProductSection ovf:class="org.apache.httpd">
1677     <Info>Product customization for the installed Apache Web Server</Info>
1678     <Product>Apache Distribution Y</Product>
1679     <Version>2.6.6</Version>
1680     <Property ovf:key="httpPort" ovf:type="uint16" ovf:value="80"
1681         ovf:userConfigurable="true">
1682         <Description>Port number for HTTP requests</Description>
1683     </Property>
1684     <Property ovf:key="httpsPort" ovf:type="uint16" ovf:value="443"
1685         ovf:userConfigurable="true">
1686         <Description>Port number for HTTPS requests</Description>
1687     </Property>
1688     <Property ovf:key="startThreads" ovf:type="uint16" ovf:value="50"
1689         ovf:userConfigurable="true">
1690         <Description>Number of threads created on startup. </Description>
1691     </Property>
1692     <Property ovf:key="minSpareThreads" ovf:type="uint16" ovf:value="15"
1693         ovf:userConfigurable="true">
1694         <Description>Minimum number of idle threads to handle request spikes.
1695         </Description>
1696     </Property>
1697     <Property ovf:key="maxSpareThreads" ovf:type="uint16" ovf:value="30"
1698         ovf:userConfigurable="true">
1699         <Description>Maximum number of idle threads </Description>
1700     </Property>
1701     <Property ovf:key="maxClients" ovf:type="uint16" ovf:value="256"
1702         ovf:userConfigurable="true">
1703         <Description>Limits the number of simultaneous requests that will be
1704         served. </Description>
1705     </Property>
1706 </ProductSection>
1707 <!-- MySQL component configuration parameters -->
1708 <ProductSection ovf:class="org.mysql.db">
1709     <Info>Product customization for the installed MySQL Database Server</Info>
1710     <Product>MySQL Distribution Z</Product>
1711     <Version>5.0</Version>
1712     <Property ovf:key="queryCacheSizeMB" ovf:type="uint16" ovf:value="32"
1713         ovf:userConfigurable="true">
1714         <Description>Buffer to cache repeated queries for faster access (in
1715         MB)</Description>
1716     </Property>
1717     <Property ovf:key="maxConnections" ovf:type="uint16" ovf:value="500"
1718         ovf:userConfigurable="true">
1719         <Description>The number of concurrent connections that can be
1720         served</Description>
1721     </Property>
1722     <Property ovf:key="waitTimeout" ovf:type="uint16" ovf:value="100"
1723         ovf:userConfigurable="true">
1724         <Description>Number of seconds to wait before timing out a connection
1725         </Description>
1726     </Property>
1727 </ProductSection>
1728 <!-- PHP component configuration parameters -->
1729 <ProductSection ovf:class="net.php">
1730     <Info>Product customization for the installed PHP component</Info>
1731     <Product>PHP Distribution U</Product>
1732     <Version>5.0</Version>
1733     <Property ovf:key="sessionTimeout" ovf:type="uint16" ovf:value="5"
1734         ovf:userConfigurable="true">
1735         <Description> How many minutes a session has to be idle before it is
1736         timed out </Description>
1737     </Property>
1738     <Property ovf:key="concurrentSessions" ovf:type="uint16" ovf:value="500"
1739         ovf:userConfigurable="true">
1740         <Description> The number of concurrent sessions that can be served
1741         </Description>
1742     </Property>
1743     <Property ovf:key="memoryLimit" ovf:type="uint16" ovf:value="32"
1744         ovf:userConfigurable="true">
1745         <Description> How much memory in megabytes a script can consume before

```

```

1746         being killed </Description>
1747     </Property>
1748 </ProductSection>
1749 <StartupSection>
1750     <Info>Startup order of the virtual machines</Info>
1751     <Item ovf:id="DbServer" ovf:order="1" ovf:startDelay="120"
1752         ovf:startAction="powerOn" ovf:waitingForGuest="true" ovf:stopDelay="120"
1753         ovf:stopAction="guestShutdown" />
1754     <Item ovf:id="AppServer" ovf:order="2" ovf:startDelay="120"
1755         ovf:startAction="powerOn" ovf:waitingForGuest="true" ovf:stopDelay="120"
1756         ovf:stopAction="guestShutdown" />
1757 </StartupSection>
1758 <VirtualSystem ovf:id="AppServer">
1759     <Info>The configuration of the AppServer virtual machine</Info>
1760     <Name>Application Server</Name>
1761     <!-- Linux component configuration parameters -->
1762     <ProductSection ovf:class="org.linuxdistx">
1763         <Info>Product customization for the installed Linux system</Info>
1764         <Product>Linux Distribution X</Product>
1765         <Version>2.6.3</Version>
1766         <Property ovf:key="hostname" ovf:type="string" ovf:value="{appHostName}" />
1767         <Property ovf:key="ip" ovf:type="string" ovf:value="{appIp}" />
1768         <Property ovf:key="subnet" ovf:type="string" ovf:value="{subnet}" />
1769         <Property ovf:key="gateway" ovf:type="string" ovf:value="{gateway}" />
1770         <Property ovf:key="dns" ovf:type="string" ovf:value="{dns}" />
1771         <Property ovf:key="netCoreRmemMaxMB" ovf:type="string"
1772             ovf:value="{netCoreRmemMaxMB}" />
1773         <Property ovf:key="netCoreWmemMaxMB" ovf:type="string"
1774             ovf:value="{netCoreWmemMaxMB}" />
1775     </ProductSection>
1776     <OperatingSystemSection ovf:id="99">
1777         <Info>Guest Operating System</Info>
1778         <Description>Linux 2.6.x</Description>
1779     </OperatingSystemSection>
1780     <VirtualHardwareSection>
1781         <Info>Virtual Hardware Requirements: 256 MB, 1 CPU, 1 disk, 1 NIC</Info>
1782         <System>
1783             <vssd:ElementName>Virtual Hardware Family</vssd:ElementName>
1784             <vssd:InstanceID>0</vssd:InstanceID>
1785             <vssd:VirtualSystemType>vmx-04</vssd:VirtualSystemType>
1786         </System>
1787         <Item>
1788             <rasd:Description>Number of virtual CPUs</rasd:Description>
1789             <rasd:ElementName>1 virtual CPU</rasd:ElementName>
1790             <rasd:InstanceID>1</rasd:InstanceID>
1791             <rasd:ResourceType>3</rasd:ResourceType>
1792             <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
1793         </Item>
1794         <Item>
1795             <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1796             <rasd:Description>Memory Size</rasd:Description>
1797             <rasd:ElementName>256 MB of memory</rasd:ElementName>
1798             <rasd:InstanceID>2</rasd:InstanceID>
1799             <rasd:ResourceType>4</rasd:ResourceType>
1800             <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
1801         </Item>
1802         <Item>
1803             <rasd:AutomaticAllocation>true</rasd:AutomaticAllocation>
1804             <rasd:Connection>VM Network</rasd:Connection>
1805             <rasd:ElementName>Ethernet adapter on "VM Network"</rasd:ElementName>
1806             <rasd:InstanceID>3</rasd:InstanceID>
1807             <rasd:ResourceSubType>PCNet32</rasd:ResourceSubType>
1808             <rasd:ResourceType>10</rasd:ResourceType>
1809         </Item>
1810         <Item>
1811             <rasd:ElementName>SCSI Controller 0 - LSI Logic</rasd:ElementName>
1812             <rasd:InstanceID>4</rasd:InstanceID>
1813             <rasd:ResourceSubType>LsiLogic</rasd:ResourceSubType>
1814             <rasd:ResourceType>6</rasd:ResourceType>
1815         </Item>
1816         <Item>
1817             <rasd:ElementName>Harddisk 1</rasd:ElementName>
1818             <rasd:HostResource>ovf:/disk/lamp-app</rasd:HostResource>
1819             <rasd:InstanceID>5</rasd:InstanceID>

```

```

1820         <rasd:Parent>4</rasd:Parent>
1821         <rasd:ResourceType>17</rasd:ResourceType>
1822     </Item>
1823 </VirtualHardwareSection>
1824 </VirtualSystem>
1825 <VirtualSystem ovf:id="DB Server">
1826     <Info>The configuration of the database virtual machine</Info>
1827     <Name>Database Server</Name>
1828     <!-- Linux component configuration parameters -->
1829     <ProductSection ovf:class="org.linuxdistx">
1830         <Info>Product customization for the installed Linux system</Info>
1831         <Product>Linux Distribution X</Product>
1832         <Version>2.6.3</Version>
1833         <Property ovf:key="hostname" ovf:type="string"
1834             ovf:value="{dbHostName}"/>
1835         <Property ovf:key="ip" ovf:type="string" ovf:value="{dbIp}"/>
1836         <Property ovf:key="subnet" ovf:type="string" ovf:value="{subnet}"/>
1837         <Property ovf:key="gateway" ovf:type="string" ovf:value="{gateway}"/>
1838         <Property ovf:key="dns" ovf:type="string" ovf:value="{dns}"/>
1839         <Property ovf:key="netCoreRmemMaxMB" ovf:type="string"
1840             ovf:value="{netCoreRmemMaxMB}"/>
1841         <Property ovf:key="netCoreWmemMaxMB" ovf:type="string"
1842             ovf:value="{netCoreWmemMaxMB}"/>
1843     </ProductSection>
1844     <OperatingSystemSection ovf:id="99">
1845         <Info>Guest Operating System</Info>
1846         <Description>Linux 2.6.x</Description>
1847     </OperatingSystemSection>
1848     <VirtualHardwareSection>
1849         <Info>Virtual Hardware Requirements: 256 MB, 1 CPU, 1 disk, 1 nic</Info>
1850     <System>
1851         <vssd:ElementName>Virtual Hardware Family</vssd:ElementName>
1852         <vssd:InstanceID>0</vssd:InstanceID>
1853         <vssd:VirtualSystemType>vmx-04</vssd:VirtualSystemType>
1854     </System>
1855     <Item>
1856         <rasd:Description>Number of virtual CPUs</rasd:Description>
1857         <rasd:ElementName>1 virtual CPU</rasd:ElementName>
1858         <rasd:InstanceID>1</rasd:InstanceID>
1859         <rasd:ResourceType>3</rasd:ResourceType>
1860         <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
1861     </Item>
1862     <Item>
1863         <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
1864         <rasd:Description>Memory Size</rasd:Description>
1865         <rasd:ElementName>256 MB of memory</rasd:ElementName>
1866         <rasd:InstanceID>2</rasd:InstanceID>
1867         <rasd:ResourceType>4</rasd:ResourceType>
1868         <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
1869     </Item>
1870     <Item>
1871         <rasd:AutomaticAllocation>true</rasd:AutomaticAllocation>
1872         <rasd:Connection>VM Network</rasd:Connection>
1873         <rasd:ElementName>Ethernet adapter on "VM Network"</rasd:ElementName>
1874         <rasd:InstanceID>3</rasd:InstanceID>
1875         <rasd:ResourceType>10</rasd:ResourceType>
1876     </Item>
1877     <Item>
1878         <rasd:ElementName>SCSI Controller 0 - LSI Logic</rasd:ElementName>
1879         <rasd:InstanceID>4</rasd:InstanceID>
1880         <rasd:ResourceSubType>LsiLogic</rasd:ResourceSubType>
1881         <rasd:ResourceType>6</rasd:ResourceType>
1882     </Item>
1883     <Item>
1884         <rasd:ElementName>Harddisk 1</rasd:ElementName>
1885         <rasd:HostResource>ovf:/disk/lamp-db</rasd:HostResource>
1886         <rasd:InstanceID>5</rasd:InstanceID>
1887         <rasd:Parent>4</rasd:Parent>
1888         <rasd:ResourceType>17</rasd:ResourceType>
1889     </Item>
1890 </VirtualHardwareSection>
1891 </VirtualSystem>
1892 </VirtualSystemCollection>
1893 </Envelope>

```

1894

1895 C Extensibility Example

1896

1897

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

1898

1899

- New section elements may be defined as part of the `Section` substitution group, and used wherever the OVF schemas allow sections to be present.

1900

1901

1902

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

1903

1904

Custom meta-data is not allowed to use OVF XML namespaces. On custom elements, a boolean `ovf:required` attribute specifies whether the information in the element is required for correct behavior or optional.

1905

1906

1907

1908

1909

1910

1911

1912

1913

1914

1915

The open content model in the OVF schemas only allows extending existing types at the end. Using XML Schema 1.0 it is not easy to allow for a more flexible open content model, due to the Unique Particle Attribution rule and the necessity of adding `xs:any` declarations everywhere in the schema. The XML Schema 1.1 draft standard contains a much more flexible open content mechanism, using `xs:openContent mode="interleave"` declarations. Future versions of the OVF specification may look at supporting this.

1916 Custom Schema

1917

1918

1919

1920

A custom XML schema defining two extension types is listed below. The first declaration defines a custom member of the OVF `Section` substitution group, while the second declaration defines a simple custom type.

1921

1922

1923

1924

1925

1926

1927

1928

1929

1930

1931

1932

1933

1934

1935

1936

1937

1938

1939

1940

1941

1942

1943

1944

1945

1946

1947

1948

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  targetNamespace="http://schemas.customextension.org/1"
  xmlns:custom="http://schemas.customextension.org/1"
  xmlns="http://schemas.customextension.org/1"
  xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  attributeFormDefault="qualified"
  elementFormDefault="qualified">

  <!-- Define a custom member of the ovf:Section substitution group -->
  <xs:element name="CustomSection" type="custom:CustomSection_Type"
substitutionGroup="ovf:Section"/>

  <xs:complexType name="CustomSection_Type">
    <xs:complexContent>
      <xs:extension base="ovf:Section_Type">
        <xs:sequence>
          <xs:element name="Data" type="xs:string"/>
        </xs:sequence>
        <xs:anyAttribute namespace="##any" processContents="lax"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- Define other simple custom type not part of ovf:Section substitution group -->
  <xs:complexType name="CustomOther_Type">
```

```

1949     <xs:sequence>
1950         <xs:element name="Data" type="xs:string"/>
1951     </xs:sequence>
1952     <xs:attribute ref="ovf:required"/>
1953     <xs:anyAttribute namespace="##any" processContents="lax"/>
1954 </xs:complexType>
1955
1956 </xs:schema >

```

1957
1958

1959 Descriptor with custom extensions

1960
1961
1962
1963

A complete OVF descriptor using the custom schema above is listed below. The descriptor validates against the OVF schema and the custom schema, but apart from extension examples the descriptor is kept minimal and is as such not useful.

1964 The descriptor contains all three extension types: a custom OVF Section element, a custom element at
1965 an extension point, and a custom attribute.
1966

```

1967 <?xml version="1.0" encoding="UTF-8"?>
1968 <Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1969     xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
1970     xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
1971 schema/2/CIM_ResourceAllocationSettingData"
1972     xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1"
1973     xmlns="http://schemas.dmtf.org/ovf/envelope/1"
1974     xmlns:custom="http://schemas.customextension.org/1">
1975
1976     <!-- Dummy References element -->
1977     <References/>
1978
1979     <!-- EXAMPLE: Optional custom OVF section element with validation against custom schema -->
1980     <custom:CustomSection ovf:required="false">
1981         <Info>Description of custom extension</Info>
1982         <custom:Data>somevalue</custom:Data>
1983     </custom:CustomSection>
1984
1985     <!-- Describes all networks used in the package -->
1986     <NetworkSection>
1987         <Info>Logical networks used in the package</Info>
1988         <!-- EXAMPLE: Optional custom attribute -->
1989         <Network ovf:name="VM Network" custom:desiredCapacity="1 Gbit/s"/>
1990         <!-- EXAMPLE: Optional custom meta-data inserted at extension point with validation
1991             against custom schema -->
1992         <custom:CustomOther xsi:type="custom:CustomOther_Type" ovf:required="false">
1993             <custom:Data>somevalue</custom:Data>
1994         </custom:CustomOther>
1995     </NetworkSection>
1996
1997     <!-- Dummy Content element -->
1998     <VirtualSystem ovf:id="Dummy">
1999         <Info>Dummy VirtualSystem</Info>
2000     </VirtualSystem>
2001 </Envelope>

```

2002

2003 The OVF environment XML schemas contain extension mechanisms matching those of the OVF
2004 envelope XML schemas, so OVF environment documents are similarly extensible.