Open Virtualization Format White Paper

Version 1.0.0

Status: Informational

Publication Date: 2/6/2009

DSP2017

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.

Table of Contents

1 Introduction

1.1 Overview

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

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

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

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

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

1.2 Virtual Appliances

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

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

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

1.3 Design Goals

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

- **Optimized for distribution**
  Supports content verification and integrity checking based on industry standard public key infrastructure, and provides a basic scheme for management of software licensing.

- **Optimized for a simple, automated user experience**
  Supports validation of the entire package and each virtual machine or meta-data component of the OVF during the installation phases of the VM lifecycle management process. It also packages with the appliance relevant user-readable descriptive information that can be use by a virtualization platform to streamline the installation experience.

- **Supports both single VM and multi-VM configurations**
  Supports both standard single VM packages, and packages containing complex, multi-tier services consisting of multiple interdependent VMs.

- **Portable VM packaging**
  OVF is virtualization platform neutral, while also enabling platform-specific enhancements to be captured. It supports the full range of virtual hard disk formats used for VMs today, and is extensible to deal with future formats that may arise. Virtual machine properties are captured concisely and accurately.

- **Vendor and platform independent**
  The OVF does not rely on the use of a specific host platform, virtualization platform, or guest operating system (within the appliance).

- **Extensible**
  OVF is immediately useful – and extensible. It is designed to be extended as the industry moves forward with the virtual appliance technology. It also supports and permits the encoding of custom meta-data to support specific vertical markets.

- **Localizable**
  Supports user visible descriptions in multiple locales, and supports localization of the interactive processes during installation of an appliance. This allows a single packaged appliance to serve multiple market opportunities.

- **Open standard**
  The OVF has arisen from the collaboration of key vendors in the industry, and will be developed as a future standard for portable virtual machines.

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

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

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

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

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

- It directly supports the configuration of multi-tier applications and the composition of virtual machines to deliver composed services.

- It permits the specification of both VM and application-level configuration.

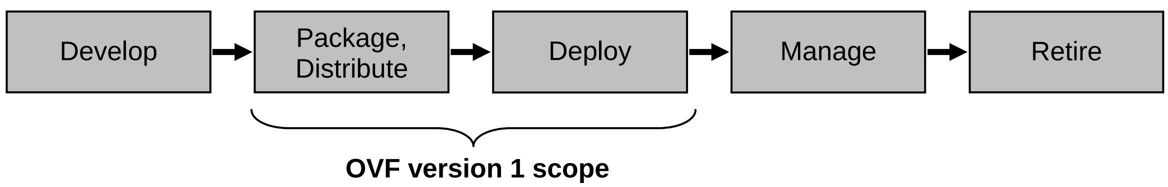
- It offers robust mechanisms for validation of the contents of the OVF, and full support for unattended installation to ease the burden of deployment for users, and thereby enhance the user's experience.

- It uses commercially accepted procedures for integrity checking of the contents of the OVF, through the use of signatures and trusted third parties. This serves to reassure the consumer of an appliance that it has not been modified since signed by the creator of the appliance. This is seen as critical to the success of the virtual appliance market, and to the viability of independent creation and online download of appliances.

- It permits commercial interests of the appliance vendor and user to be respected, by providing a basic method for presentation and acknowledgement of licensing terms associated with the appliance.

1.4 Virtual Appliance Life-Cycle

The software life cycle for virtual appliances is shown below:



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

2 Portable Virtualization Format

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

2.1 OVF Package

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

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

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

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

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

2.2 OVF Environment

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

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

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

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

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

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

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

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

2.3 Sample OVF Descriptor

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

```
<?xml version="1.0" encoding="UTF-8"?>
<Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://schemas.dmtf.org/ovf/1/envelope"
    xmlns:ovf="http://schemas.dmtf.org/ovf/1/envelope"
    xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
    xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
schema/2/CIM_ResourceAllocationSettingData">

    <!-- References to all external files -->
    <References>
        <File ovf:id="file1" ovf:href="vmdisk1.vmdk" ovf:size="180114671"/>
    </References>
    <!-- Describes meta-information for all virtual disks in the package -->
    <DiskSection>
        <Info>Describes the set of virtual disks</Info>
        <Disk ovf:diskId="vmdisk1" ovf:fileRef="file1" ovf:capacity="4294967296"
              ovf:format="http://www.vmware.com/interfaces/specifications/vmdk.html#sparse"/>
    </DiskSection>
    <!-- Describes all networks used in the package -->
    <NetworkSection>
        <Info>List of logical networks used in the package</Info>
        <Network ovf:name="VM Network">
            <Description>The network that the service will be available on</Description>
        </Network>
    </NetworkSection>
    <VirtualSystem ovf:id="vm">
        <Info>Describes a virtual machine</Info>
        <Name>Virtual Appliance One</Name>
        <ProductSection>
            <Info>Describes product information for the appliance</Info>
            <Product>The Great Appliance</Product>
            <Vendor>Some Great Corporation</Vendor>
            <Version>13.00</Version>
            <FullVersion>13.00-b5</FullVersion>
            <ProductUrl>http://www.somegreatcorporation.com/greatappliance</ProductUrl>
            <VendorUrl>http://www.somegreatcorporation.com/</VendorUrl>
            <Property ovf:key="admin.email" ovf:type="string">
                <Description>Email address of administrator</Description>
            </Property>
            <Property ovf:key="app.ip" ovf:type="string" ovf:defaultValue="192.168.0.10">
                <Description>The IP address of this appliance</Description>
            </Property>
        </ProductSection>
        <AnnotationSection ovf:required="false">
            <Info>A random annotation on this service. It can be ignored</Info>
            <Annotation>Contact customer support if you have any problems</Annotation>
        </AnnotationSection>
         <EulaSection>
            <Info>License information for the appliance</Info>
            <License>Insert your favorite license here</License>
        </EulaSection>
        <VirtualHardwareSection>
            <Info>256MB, 1 CPU, 1 disk, 1 nic</Info>
            <Item>
```

```
                <rasd:Description>Number of virtual CPUs</rasd:Description>
                <rasd:ElementName>1 virtual CPU</rasd:ElementName>
                <rasd:InstanceID>1</rasd:InstanceID>
                <rasd:ResourceType>3</rasd:ResourceType>
                <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
            </Item>
            <Item>
                <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
                <rasd:Description>Memory Size</rasd:Description>
                <rasd:ElementName>256 MB of memory</rasd:ElementName>
                <rasd:InstanceID>2</rasd:InstanceID>
                <rasd:ResourceType>4</rasd:ResourceType>
                <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
            </Item>
            <Item>
                <rasd:AutomaticAllocation>true</rasd:AutomaticAllocation>
                <rasd:Connection>VM Network</rasd:Connection>
                <rasd:ElementName>Ethernet adapter on "VM Network"</rasd:ElementName>
                <rasd:InstanceID>4000</rasd:InstanceID>
                <rasd:ResourceType>10</rasd:ResourceType>
             </Item>
            <Item>
                <rasd:ElementName>Harddisk 1</rasd:ElementName>
                <rasd:HostResource>ovf:/disk/vmdisk1</rasd:HostResource>
                <rasd:InstanceID>22001</rasd:InstanceID>
                <rasd:ResourceType>17</rasd:ResourceType>
            </Item>
        </VirtualHardwareSection>
        <OperatingSystemSection ovf:id="58" ovf:required="false">
          <Info>Guest Operating System</Info>
          <Description>Windows 2000 Advanced Server</Description>
        </OperatingSystemSection>
    </VirtualSystem>
</Envelope>
```

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

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

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

3 Using the Open Virtualization Format

3.1 Creation

The creation of an OVF involves the i) packaging of a set of VMs onto a set of virtual disks, ii) appropriately encoding those virtual disks, iii) attaching an OVF descriptor with a specification of the virtual hardware, licensing, and other customization metadata, and iv) optionally digitally signing the package. The process of installing or importing an OVF occurs when a virtualization platform consumes the OVF and creates a set of virtual machines from its contents.

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

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

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

3.2 Deployment

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

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

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

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

4 Features

4.1 Virtual Hardware Description

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

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

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

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

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

```
<VirtualHardwareSection>
  <Info>500Mb, 1 CPU, 1 disk, 1 nic virtual machine</Info>
     <System>
        ...
     </System>
     <Item>
       ...
     </Item>
     ...
</VirtualHardwareSection>
<VirtualHardwareSection>
  <Info>500Mb, 1 CPU, 1 disk, 1 nic virtual machine</Info>
     <System>
        ...
     </System>
     <Item>
       ...
     </Item>
     ...
</VirtualHardwareSection>
```

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

Example where specific device types are specified:

```
<Item>
    <rasd:ElementName>SCSI Controller 0</rasd:ElementName>
    <rasd:InstanceID>1000</rasd:InstanceID>
    <rasd:ResourceSubType>LsiLogic BusLogic</rasd:ResourceSubType>
    <rasd:ResourceType>6</rasd:ResourceType>
</Item>
<Item>
    <rasd:ElementName>Harddisk 1</rasd:ElementName>
    <rasd:HostResource>ovf:/disk/vmdisk1</rasd:HostResource>
    <rasd:InstanceID>22001</rasd:InstanceID>
    <rasd:Parent>1000</rasd:Parent>
    <rasd:ResourceType>17</rasd:ResourceType>
</Item>
```

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

4.2 Deployment Options

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

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

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

Example list of configurations:

```
<DeploymentOptionSection>
    <Configuration ovf:id="min">
        <Label>Minimal</Label>
        <Description>Minimal setup</Description>
    </Configuration>
    <Configuration ovf:id="normal" ovf:default="yes">
        <Label>Normal</Label>
        <Description>Standard setup</Description>
    </Configuration>
    ... more configurations ...
</DeploymentOptionSection>
```

Resource requirement example:

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

```
        ... normal configuration ...
   </Item>
   <Item ovf:configuration="min">
        ... overwrites for minimal configuration ...
   </Item>
</ResourceAllocationSection>
```

`VirtualHardwareSection` example:

```
<VirtualHardwareSection>
  <Info>...</Info>
  <Item>
      <rasd:AllocationUnits>hertz * 10^6</rasd:AllocationUnits>
      <rasd:ElementName>1 CPU and 500 MHz reservation</rasd:ElementName>
      <rasd:InstanceID>1</rasd:InstanceID>
      <rasd:Reservation>500</rasd:Reservation>
      <rasd:ResourceType>4</rasd:ResourceType>
      <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
  </Item>
  ...
  <Item ovf:configuration="big">
      <rasd:ElementName>1 CPU and 800 MHz reservation</rasd:ElementName>
      <rasd:InstanceID>0</rasd:InstanceID>
      <rasd:Reservation>600</rasd:Reservation>
      <rasd:ResourceType>3</rasd:ResourceType>
  </Item>
</VirtualHardwareSection>
```

4.3 Deployment Customization

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

```
<ProductSection>
     <Info>Describes product information for the service</Info>
     <Product>MyService Web Portal</Product>
     <Vendor>Some Random Organization</Vendor>
     <Version>4.5</Version>
     <FullVersion>4.5-b4523</FullVersion>
     <ProductUrl>http://www.vmware.com/go/ovf</ProductUrl>
     <VendorUrl>http://www.vmware.com/</VendorUrl>
     <Property ovf:key="adminEmail" ovf:type="string" ovf:userConfigurable="true">
        <Description>Email address of administrator</Description>
     </Property>
     <Property ovf:key="appIp" ovf:type="string" ovf:userConfigurable="true">
         <Description>IP address of the application</Description>
    </Property>
</ProductSection>
```

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

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

4.4 Internationalization

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

```
<Envelope ...>
   ...
  <Info ovf:msgid="info.os">Operating System</Info>
   ...
  <Strings xml:lang="da-DA">
```

```
    <Msg ovf:msgid="info.os">Operativsystem</Msg>
    ...
  </Strings>
  <Strings xml:lang="de-DE">
    <Msg ovf:msgid="info.os">Betriebssystem</Msg>
    ...
  </Strings>
</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:

```
<Envelope ...>
   <References>
      ...
      <File ovf:id="da-DA-resources" ovf:href="german.msg"/>
      <File ovf:id="de-DE-resources" ovf:href="danish.msg"/>
      ...
   </References>
      ...
   <Strings xml:lang="da-DA" ovf:fileRef="da-da-resources"/>
   <Strings xml:lang="de-DE" ovf:fileRef="de-de-resources"/>
</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:

```
<ns:BuildInformationSection ovf:required=”false”>
      <Info>Specifies information on how a virtual machine was created</Info>
      <BuildNumber> ... </BuildNumber >
      <BuildDate> ... </BuildDate >
      <BuildSystem> ... </BuildSystem>
          ...
</ns:BuildInformationSection>
```

Example of extending existing section:

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

- OVF descriptor only contains meta-data defined in the OVF specification, i.e. no custom extensions are present.
  Conformance Level: 1.

- OVF descriptor contains meta-data with custom extensions, but all such extensions are optional.
  Conformance Level: 2.

- OVF descriptor contains meta-data with custom extensions, and at least one such extension is required.
  Conformance Level: 3.

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

5 Portability

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

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

- The VMs in the OVF could contain virtual disks in a format that is not understood by the hypervisor attempting the installation. While it is reasonable to expect that most hypervisors will be able to import and/or export VMs in any of the major virtual hard disk formats, newer formats may arise that are supported by the OVF and not a particular hypervisor. It may be useful in future versions of this specification, to stipulate a required set of virtual hard disk formats that must be supported by an OVF compliant hypervisor.

- The installed guest software may not support the virtual hardware presented by the hypervisor. By way of example, the Xen hypervisor does not by default offer a virtualized floppy disk device to guests. One could conceive of a guest VM that would require interaction with a floppy disk controller and which therefore would not be able to execute the VM correctly.

- The installed guest software does not support the CPU architecture. For example, the guest software might execute CPU operations specific to certain processor models or require specific floating point support, or contain opcodes specific to a particular vendor's CPU.

- The virtualization platform might not understand a feature requested in the OVF descriptor. For example, composed services may not be supported. Since the OVF standard will evolve independently of virtualization products, at any point an OVF might be unsupportable on a virtualization platform that pre-dates that OVF specification.

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

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

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

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

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

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

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

6 Future Versions of the OVF Specification

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

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

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

7 Conclusion

The OVF specification offers a portable virtual appliance format that is intended for broad adoption across the IT industry. The OVF specification is intended to be immediately useful, to solve an immediate business need, and to facilitate the rapid adoption of a common, backwards compatible, yet rich virtual machine format. OVF is complementary to existing IT management standards and frameworks, and will be further developed within a standards organization. OVF promotes customer confidence through the collaborative development of common standards for portability and interchange of virtual machines between different vendors' virtualization platforms, and promotes best-of-breed competition through its openness and extensibility.

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

A Multi-tiered Petstore Example

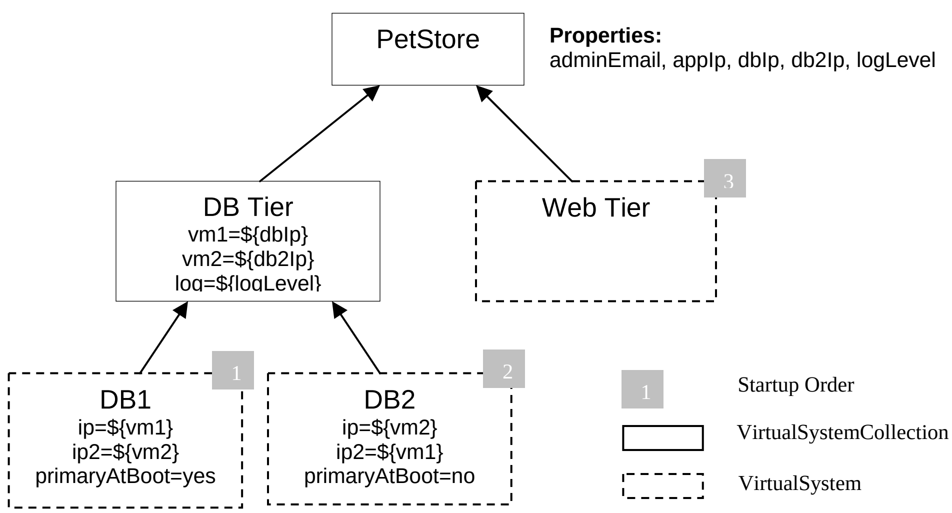
This example will demonstrate several advanced OVF concepts:

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

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

Architecture and Packaging

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



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

Properties

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

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

Web Guest Software:

Property	Description
appIp	IP address of the WebServer.
dbIp	IP address of the database server to connect to.
adminEmail	Email address for support
logLevel	Logging level

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

Database Guest Software:

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

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

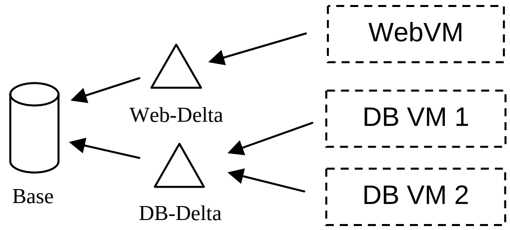
The database software is an off-the-shelf software package and the vendor has chosen the "com.mydb.db" as the unique name for all the properties. This can be seen in the OVF descriptor with the inclusion of the ovf:class attribute on the ProductSection.

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

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

Disk Layout

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



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

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

Complete OVF Descriptor

```
<?xml version="1.0" encoding="UTF-8"?>
<Envelope
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://schemas.dmtf.org/ovf/envelope/1"
    xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1"
    xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
    xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
schema/2/CIM_ResourceAllocationSettingData"
    <!-- References to all external files -->
    <References>
        <File ovf:id="base" ovf:href="base.vmdk" ovf:size="180114671"/>
        <File ovf:id="webdelta" ovf:href="webapp-delta.vmdk" ovf:size="123413"/>
        <File ovf:id="dbdelta" ovf:href="dbapp-delta.vmdk" ovf:size="343243"/>
    </References>
    <!-- Describes meta-information about all virtual disks in the package.
        This example is encoded as a delta-disk hierarchy.
     -->
    <DiskSection>
        <Info>Describes the set of virtual disks</Info>
        <Disk ovf:diskId="base" ovf:fileRef="base" ovf:capacity="4294967296"
            ovf:populatedSize="1924967692"
            ovf:format="http://www.vmware.com/specifications/vmdk.html#streamOptimized"/>
        <Disk ovf:diskId="web" ovf:fileRef="webappdelta" ovf:parentRef="base"
            ovf:capacity="4294967296"
            ovf:format="http://www.vmware.com/specifications/vmdk.html#streamOptimized"/>
        <Disk ovf:diskId="db" ovf:fileRef="dbdelta" ovf:parentRef="base"
            ovf:capacity="4294967296"
            ovf:format="http://www.vmware.com/specifications/vmdk.html#streamOptimized"/>
    </DiskSection>
    <!-- Describes all networks used in the package -->
    <NetworkSection>
        <Info>List of logical networks used in the package</Info>
        <Network ovf:name="VM Network">
            <Description ovf:msgid="network.description">The network that the service
                will be available on</Description>
        </Network>
    </NetworkSection>
    <!-- Deployment options for the packages -->
    <DeploymentOptionSection>
        <Info>List of deployment options available in the package</Info>
        <Configuration ovf:id="minimal">
            <Label ovf:msgid="minimal.label">Minimal</Label>
```

```
            <Description ovf:msgid="minimal.description">Deploy service with minimal
                resource use</Description>
        </Configuration>
        <Configuration ovf:id="standard" ovf:default="true">
            <Label ovf:msgid="standard.label">Standard</Label>
            <Description ovf:msgid="standard.description">Deploy service with standard
                resource use</Description>
        </Configuration>
    </DeploymentOptionSection>
    <!-- PetStore Virtual System Collection -->
    <VirtualSystemCollection ovf:id="PetStore">
        <Info>The packaging of the PetStoreService multi-tier application</Info>
        <Name>PetStore Service</Name>
        <!-- Overall information about the product -->
        <ProductSection>
            <Info>Describes product information for the service</Info>
            <Product>PetStore Web Portal</Product>
            <Vendor>Some Random Organization</Vendor>
            <Version>4.5</Version>
            <FullVersion>4.5-b4523</FullVersion>
            <ProductUrl>http://www.vmware.com/go/ovf</ProductUrl>
            <VendorUrl>http://www.vmware.com/</VendorUrl>
            <Category ovf:msgid="category.email">Email properties</Category>
            <Property ovf:key="adminEmail" ovf:type="string" ovf:userConfigurable="true">
                <Label ovf:msgid="property.email.label">Admin email</Label>
                <Description ovf:msgid="property.email.description">Email address of
                    service administrator</Description>
            </Property>
            <Category ovf:msgid="category.network">Network properties</Category>
            <Property ovf:key="appIp" ovf:type="string"
                ovf:userConfigurable="true">
                <Label ovf:msgid="property.appip.label">IP</Label>
                <Description ovf:msgid="property.appip.description">IP address of the
                    service</Description>
            </Property>
            <Property ovf:key="dbIp" ovf:type="string" ovf:userConfigurable="true">
                <Label ovf:msgid="property.dpip.label">IP for DB</Label>
                <Description ovf:msgid="property.dpip.description">Primary IP address of
                    the database</Description>
            </Property>
            <Property ovf:key="db2Ip" ovf:type="string"
                ovf:userConfigurable="true">
                <Label ovf:msgid="property.dpip2.label">IP for DB2</Label>
                <Description ovf:msgid="property.dpip2.description">A secondary IP
                    address for the database</Description>
            </Property>
            <Category ovf:msgid="category.logging">Logging properties</Category>
            <Property ovf:key="logLevel" ovf:type="string" ovf:value="normal"
                ovf:userConfigurable="true">
                <Label ovf:msgid="property.loglevel.label">Loglevel</Label>
                <Description ovf:msgid="property.loglevel.description">Logging level for
                    the service</Description>
                <Value ovf:value="low" ovf:configuration="minimal"/>
            </Property>
        </ProductSection>
        <AnnotationSection ovf:required="false">
            <Info>A annotation on this service</Info>
            <Annotation ovf:msgid="annotation.annotation">Contact customer support for
                any urgent issues</Annotation>
        </AnnotationSection>
        <ResourceAllocationSection ovf:required="false">
            <Info>Defines minimum reservations for CPU and memory</Info>
            <Item>
                <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
                <rasd:ElementName>512 MB reservation</rasd:ElementName>
                <rasd:InstanceID>0</rasd:InstanceID>
                <rasd:Reservation>512</rasd:Reservation>
                <rasd:ResourceType>4</rasd:ResourceType>
            </Item>
            <Item ovf:configuration="minimal">
                <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
                <rasd:ElementName>384 MB reservation</rasd:ElementName>
                <rasd:InstanceID>0</rasd:InstanceID>
                <rasd:Reservation>384</rasd:Reservation>
```

```
                <rasd:ResourceType>4</rasd:ResourceType>
            </Item>
            <Item>
                <rasd:AllocationUnits>MHz</rasd:AllocationUnits>
                <rasd:ElementName>1000 MHz reservation</rasd:ElementName>
                <rasd:InstanceID>1</rasd:InstanceID>
                <rasd:Reservation>500</rasd:Reservation>
                <rasd:ResourceType>3</rasd:ResourceType>
            </Item>
            <Item ovf:bound="min">
                <rasd:AllocationUnits>MHz</rasd:AllocationUnits>
                <rasd:ElementName>500 MHz reservation</rasd:ElementName>
                <rasd:InstanceID>1</rasd:InstanceID>
                <rasd:Reservation>500</rasd:Reservation>
                <rasd:ResourceType>3</rasd:ResourceType>
            </Item>
            <Item ovf:bound="max">
                <rasd:AllocationUnits>MHz</rasd:AllocationUnits>
                <rasd:ElementName>1500 MHz reservation</rasd:ElementName>
                <rasd:InstanceID>1</rasd:InstanceID>
                <rasd:Reservation>1500</rasd:Reservation>
                <rasd:ResourceType>3</rasd:ResourceType>
            </Item>
        </ResourceAllocationSection>
        <StartupSection>
            <Info>Specifies how the composite service is powered-on and off</Info>
            <Item ovf:id="DBTier" ovf:order="1" ovf:startDelay="120"
                ovf:startAction="powerOn" ovf:waitingForGuest="true" ovf:stopDelay="120"
                ovf:stopAction="guestShutdown"/>
            <Item ovf:id="WebTier" ovf:order="2" ovf:startDelay="120"
                ovf:startAction="powerOn" ovf:waitingForGuest="true" ovf:stopDelay="120"
                ovf:stopAction="guestShutdown"/>
        </StartupSection>
        <VirtualSystem ovf:id="WebTier">
            <Info>The virtual machine containing the WebServer application</Info>
            <ProductSection>
                <Info>Describes the product information</Info>
                <Product>Apache Webserver</Product>
                <Vendor>Apache Software Foundation</Vendor>
                <Version>6.5</Version>
                <FullVersion>6.5-b2432</FullVersion>
            </ProductSection>
            <OperatingSystemSection ovf:id="97">
                <Info>Guest Operating System</Info>
                <Description>Linux 2.4.x</Description>
            </OperatingSystemSection>
            <VirtualHardwareSection>
                <Info>256 MB, 1 CPU, 1 disk, 1 nic virtual machine</Info>
                <System>
                    <vssd:ElementName>Virtual Hardware Family</vssd:ElementName>
                    <vssd:InstanceID>0</vssd:InstanceID>
                    <vssd:VirtualSystemType>vmx-04</vssd:VirtualSystemType>
                </System>
                <Item>
                    <rasd:Description>Number of virtual CPUs</rasd:Description>
                    <rasd:ElementName>1 virtual CPU</rasd:ElementName>
                    <rasd:InstanceID>1</rasd:InstanceID>
                    <rasd:ResourceType>3</rasd:ResourceType>
                    <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
                </Item>
                <Item>
                    <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
                    <rasd:Description>Memory Size</rasd:Description>
                    <rasd:ElementName>256 MB of memory</rasd:ElementName>
                    <rasd:InstanceID>2</rasd:InstanceID>
                    <rasd:ResourceType>4</rasd:ResourceType>
                    <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
                </Item>
                <Item>
                    <rasd:AutomaticAllocation>true</rasd:AutomaticAllocation>
                    <rasd:Connection>VM Network</rasd:Connection>
                    <rasd:ElementName>Ethernet adapter on "VM Network"</rasd:ElementName>
                    <rasd:InstanceID>3</rasd:InstanceID>
                    <rasd:ResourceSubType>PCNet32</rasd:ResourceSubType>
```

```
                    <rasd:ResourceType>10</rasd:ResourceType>
                </Item>
                <Item>
                    <rasd:AddressOnParent>1</rasd:AddressOnParent>
                    <rasd:ElementName>SCSI Controller 0 - LSI Logic</rasd:ElementName>
                    <rasd:InstanceID>1000</rasd:InstanceID>
                    <rasd:ResourceSubType>LsiLogic</rasd:ResourceSubType>
                    <rasd:ResourceType>6</rasd:ResourceType>
                </Item>
                <Item>
                    <rasd:AddressOnParent>0</rasd:AddressOnParent>
                    <rasd:ElementName>Harddisk 1</rasd:ElementName>
                    <rasd:HostResource>ovf:/disk/web</rasd:HostResource>
                    <rasd:InstanceID>22001</rasd:InstanceID>
                    <rasd:Parent>1000</rasd:Parent>
                    <rasd:ResourceType>17</rasd:ResourceType>
                </Item>
            </VirtualHardwareSection>
        </VirtualSystem>
        <!-- Database Tier -->
        <VirtualSystemCollection ovf:id="DBTier">
            <Info>Describes a clustered database instance</Info>
            <ProductSection ovf:class="com.mydb.db">
                <Info>Product Information</Info>
                <Product>Somebody Clustered SQL Server</Product>
                <Vendor>TBD</Vendor>
                <Version>2.5</Version>
                <FullVersion>2.5-b1234</FullVersion>
                <Property ovf:key="vm1" ovf:value="${dbIp}" ovf:type="string"/>
                <Property ovf:key="vm2" ovf:value="${db2Ip} " ovf:type="string"/>
                <Property ovf:key="log" ovf:value="${logLevel}" ovf:type="string"/>
            </ProductSection>
            <StartupSection>
                <Info>Specifies how the composite service is powered-on and off</Info>
                <Item ovf:id="DB1" ovf:order="1" ovf:startDelay="120"
                    ovf:startAction="powerOn" ovf:waitingForGuest="true"
                    ovf:stopDelay="120" ovf:stopAction="guestShutdown"/>
                <Item ovf:id="DB2" ovf:order="2" ovf:startDelay="120"
                    ovf:startAction="powerOn" ovf:waitingForGuest="true"
                    ovf:stopDelay="120" ovf:stopAction="guestShutdown"/>
            </StartupSection>
            <!-- DB VM 1 -->
            <VirtualSystem ovf:id="DB1">
                <Info>Describes a virtual machine with the database image installed</Info>
                <Name>Database Instance I</Name>
                <ProductSection ovf:class="com.mydb.db">
                    <Info>Specifies the OVF properties available in the OVF environment</Info>
                    <Property ovf:key="ip" ovf:value="${vm1}" ovf:type="string"/>
                    <Property ovf:key="ip2" ovf:value="${vm2} " ovf:type="string"/>
                    <Property ovf:key="primaryAtBoot" ovf:value="yes" ovf:type="string"/>
                </ProductSection>
                <VirtualHardwareSection>
                    <Info>256 MB, 1 CPU, 1 disk, 1 nic virtual machine</Info>
                    <System>
                        <vssd:ElementName>Virtual Hardware Family</vssd:ElementName>
                        <vssd:InstanceID>0</vssd:InstanceID>
                        <vssd:VirtualSystemType>vmx-04</vssd:VirtualSystemType>
                    </System>
                    <Item>
                        <rasd:Description>Number of virtual CPUs</rasd:Description>
                        <rasd:ElementName>1 virtual CPU</rasd:ElementName>
                        <rasd:InstanceID>1</rasd:InstanceID>
                        <rasd:ResourceType>3</rasd:ResourceType>
                        <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
                    </Item>
                    <Item>
                        <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
                        <rasd:Description>Memory Size</rasd:Description>
                        <rasd:ElementName>256 MB of memory</rasd:ElementName>
                        <rasd:InstanceID>2</rasd:InstanceID>
                        <rasd:ResourceType>4</rasd:ResourceType>
                        <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
                    </Item>
                    <Item>
```

```xml
                        <rasd:AutomaticAllocation>true</rasd:AutomaticAllocation>
                        <rasd:Connection>VM Network</rasd:Connection>
                        <rasd:ElementName>Ethernet adapter on "VM Network"</rasd:ElementName>
                        <rasd:InstanceID>3</rasd:InstanceID>
                        <rasd:ResourceSubType>PCNet32</rasd:ResourceSubType>
                        <rasd:ResourceType>10</rasd:ResourceType>
                    </Item>
                    <Item>
                        <rasd:AddressOnParent>1</rasd:AddressOnParent>
                        <rasd:ElementName>SCSI Controller 0 - LSI Logic</rasd:ElementName>
                        <rasd:InstanceID>1000</rasd:InstanceID>
                        <rasd:ResourceSubType>LsiLogic</rasd:ResourceSubType>
                        <rasd:ResourceType>6</rasd:ResourceType>
                    </Item>
                    <Item>
                        <rasd:AddressOnParent>0</rasd:AddressOnParent>
                        <rasd:ElementName>Harddisk 1</rasd:ElementName>
                        <rasd:HostResource>ovf:/disk/db</rasd:HostResource>
                        <rasd:InstanceID>22001</rasd:InstanceID>
                        <rasd:Parent>1000</rasd:Parent>
                        <rasd:ResourceType>17</rasd:ResourceType>
                    </Item>
                </VirtualHardwareSection>
                <OperatingSystemSection ovf:id="97">
                    <Info>Guest Operating System</Info>
                    <Description>Linux 2.4.x</Description>
                </OperatingSystemSection>
            </VirtualSystem>
            <!-- DB VM 2 -->
            <VirtualSystem ovf:id="DB2">
                <Info>Describes a virtual machine with the database image installed</Info>
                <Name>Database Instance II</Name>
                <ProductSection ovf:class="com.mydb.db">
                    <Info>Specifies the OVF properties available in the OVF environment</Info>
                    <Property ovf:key="ip" ovf:value="${vm2}" ovf:type="string"/>
                    <Property ovf:key="ip2" ovf:value="${vm1} " ovf:type="string"/>
                    <Property ovf:key="primaryAtBoot" ovf:value="no" ovf:type="string"/>
                </ProductSection>
                <VirtualHardwareSection>
                    <Info>256 MB, 1 CPU, 1 disk, 1 nic virtual machine</Info>
                    <System>
                        <vssd:ElementName>Virtual Hardware Family</vssd:ElementName>
                        <vssd:InstanceID>0</vssd:InstanceID>
                        <vssd:VirtualSystemType>vmx-04</vssd:VirtualSystemType>
                    </System>
                    <Item>
                        <rasd:Description>Number of virtual CPUs</rasd:Description>
                        <rasd:ElementName>1 virtual CPU</rasd:ElementName>
                        <rasd:InstanceID>1</rasd:InstanceID>
                        <rasd:ResourceType>3</rasd:ResourceType>
                        <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
                    </Item>
                    <Item>
                        <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
                        <rasd:Description>Memory Size</rasd:Description>
                        <rasd:ElementName>256 MB of memory</rasd:ElementName>
                        <rasd:InstanceID>2</rasd:InstanceID>
                        <rasd:ResourceType>4</rasd:ResourceType>
                        <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
                    </Item>
                    <Item>
                        <rasd:AutomaticAllocation>true</rasd:AutomaticAllocation>
                        <rasd:Connection>VM Network</rasd:Connection>
                        <rasd:ElementName>Ethernet adapter on "VM Network"</rasd:ElementName>
                        <rasd:InstanceID>3</rasd:InstanceID>
                        <rasd:ResourceSubType>PCNet32</rasd:ResourceSubType>
                        <rasd:ResourceType>10</rasd:ResourceType>
                    </Item>
                    <Item>
                        <rasd:AddressOnParent>1</rasd:AddressOnParent>
                        <rasd:ElementName>SCSI Controller 0 - LSI Logic</rasd:ElementName>
                        <rasd:InstanceID>1000</rasd:InstanceID>
                        <rasd:ResourceSubType>LsiLogic</rasd:ResourceSubType>
                        <rasd:ResourceType>6</rasd:ResourceType>
```

```
                    </Item>
                    <Item>
                        <rasd:AddressOnParent>0</rasd:AddressOnParent>
                        <rasd:ElementName>Harddisk 1</rasd:ElementName>
                        <rasd:HostResource>ovf:/disk/db</rasd:HostResource>
                        <rasd:InstanceID>22001</rasd:InstanceID>
                        <rasd:Parent>1000</rasd:Parent>
                        <rasd:ResourceType>17</rasd:ResourceType>
                    </Item>
                </VirtualHardwareSection>
                <OperatingSystemSection ovf:id="97">
                    <Info>Guest Operating System</Info>
                    <Description>Linux 2.4.x</Description>
                </OperatingSystemSection>
            </VirtualSystem>
        </VirtualSystemCollection>
    </VirtualSystemCollection>
    <!-- External I18N bundles -->
    <Strings xml:lang="de-DE" ovf:fileRef="de-DE-bundle.xml"/>
    <!-- EmbeddedI18N bundles -->
    <Strings xml:lang="da-DA">
        <Msg ovf:msgid="network.description">Netværket servicen skal være tilgængelig på</Msg>
        <Msg ovf:msgid="annotation.annotation">Kontakt kundeservice i tilfælde af
            kritiske problemer</Msg>
        <Msg ovf:msgid="property.email.description">Email adresse for administrator</Msg>
        <Msg ovf:msgid="property.appip.description">IP adresse for service</Msg>
        <Msg ovf:msgid="property.dpip">Primær IP adresse for database</Msg>
        <Msg ovf:msgid="property.dpip2.description">Sekundær IP adresse for database</Msg>
        <Msg ovf:msgid="property.loglevel.description">Logningsniveau for service</Msg>
        <Msg ovf:msgid="minimal.label">Minimal</Msg>
        <Msg ovf:msgid="minimal.description">Installer service med minimal brug af
            resourcer</Msg>
        <Msg ovf:msgid="standard.label">Normal</Msg>
        <Msg ovf:msgid="standard.description">Installer service med normal brug af
            resourcer</Msg>
    </Strings>
</Envelope>
```

Complete OVF Environments

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

OVF environment for the WebTier virtual machine:

```
<?xml version="1.0" encoding="UTF-8"?>
<Environment
    xmlns="http://schemas.dmtf.org/ovf/environment/1"
    xmlns:ovfenv="http://schemas.dmtf.org/ovf/environment/1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    ovfenv:id="WebTier">

    <!-- Information about hypervisor platform -->
    <PlatformSection>
        <Kind>ESX Server</Kind>
        <Version>3.0.1</Version>
        <Vendor>VMware, Inc.</Vendor>
        <Locale>en_US</Locale>
    </PlatformSection>

    <!--- Properties defined for this virtual machine -->
    <PropertySection>
        <Property ovfenv:key="adminEmail" ovfenv:value="ovf-admin@vmware.com"/>
        <Property ovfenv:key="appIp" ovfenv:value="10.20.132.101"/>
        <Property ovfenv:key="dbIp" ovfenv:value="10.20.132.102"/>
        <Property ovfenv:key="db2Ip" ovfenv:value="10.20.132.103"/>
        <Property ovfenv:key="logLevel" ovfenv:value="warning"/>
    </PropertySection>

```

```
    <Entity ovfenv:id="DBTier">
        <PropertySection>
            <Property ovfenv:key="adminEmail" ovfenv:value="ovf-admin@vmware.com"/>
            <Property ovfenv:key="appIp" ovfenv:value="10.20.132.101"/>
            <Property ovfenv:key="dbIp" ovfenv:value="10.20.132.102"/>
            <Property ovfenv:key="db2Ip" ovfenv:value="10.20.132.103"/>
            <Property ovfenv:key="logLevel" ovfenv:value="warning"/>
            <Property ovfenv:key="com.mydb.db.vm1" ovfenv:value="10.20.132.102"/>
            <Property ovfenv:key="com.mydb.db.vm2" ovfenv:value="10.20.132.103"/>
            <Property ovfenv:key="com.mydb.db.log" ovfenv:value="warning"/>
        </PropertySection>
    </Entity>
</Environment>
```

OVF environment for the DB1 virtual machine:

```
<?xml version="1.0" encoding="UTF-8"?>
<Environment
    xmlns="http://schemas.dmtf.org/ovf/environment/1"
    xmlns:ovfenv="http://schemas.dmtf.org/ovf/environment/1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    ovfenv:id="DB1">

    <!-- Information about hypervisor platform -->
    <PlatformSection>
        <Kind>ESX Server</Kind>
        <Version>3.0.1</Version>
        <Vendor>VMware, Inc.</Vendor>
        <Locale>en_US</Locale>
    </PlatformSection>

    <!--- Properties defined for this virtual machine -->
    <PropertySection>
        <Property ovfenv:key="com.mydb.db.vm1" ovfenv:value="10.20.132.102"/>
        <Property ovfenv:key="com.mydb.db.vm2" ovfenv:value="10.20.132.103"/>
        <Property ovfenv:key="com.mydb.db.log" ovfenv:value="warning"/>
        <Property ovfenv:key="com.mydb.db.ip" ovfenv:value="10.20.132.102"/>
        <Property ovfenv:key="com.mydb.db.ip2" ovfenv:value="10.20.132.103"/>
        <Property ovfenv:key="com.mydb.db.primaryAtBoot" ovfenv:value="yes"/>
    </PropertySection>

    <Entity ovfenv:id="DB2">
        <PropertySection>
            <Property ovfenv:key="com.mydb.db.vm1" ovfenv:value="10.20.132.102"/>
            <Property ovfenv:key="com.mydb.db.vm2" ovfenv:value="10.20.132.103"/>
            <Property ovfenv:key="com.mydb.db.log" ovfenv:value="warning"/>
            <Property ovfenv:key="com.mydb.db.ip" ovfenv:value="10.20.132.103"/>
            <Property ovfenv:key="com.mydb.db.ip2" ovfenv:value="10.20.132.102"/>
            <Property ovfenv:key="com.mydb.db.primaryAtBoot" ovfenv:value="no"/>
        </PropertySection>
    </Entity>
</Environment>
```

B LAMP Stack Example

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

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

Deployment-time Customization

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

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

Product	Property	Description
Linux	*hostname*	Network identity of the application, including IP address.
	ip	
	subnet	
	gateway	
	dns	
	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	

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

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

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

```
    <Version>2.6.6</Version>
    <Property ovf:key="httpPort" ovf:type="uint16" ovf:value="80"
              ovf:userConfigurable="true">
        <Description>Port number for HTTP requests</Description>
    </Property>
    <Property ovf:key="httpsPort" ovf:type="uint16" ovf:value="443"
              ovf:userConfigurable="true">
        <Description>Port number for HTTPS requests</Description>
    </Property>
    <Property ovf:key="startThreads" ovf:type="uint16" ovf:value="50"
              ovf:userConfigurable="true">
        <Description>Number of threads created on startup. </Description>
    </Property>
    <Property ovf:key="minSpareThreads" ovf:type="uint16" ovf:value="15"
              ovf:userConfigurable="true">
        <Description> Minimum number of idle threads to handle request spikes.</Description>
    </Property>
    <Property ovf:key="maxSpareThreads" ovf:type="uint16" ovf:value="30"
              ovf:userConfigurable="true">
        <Description>Maximum number of idle threads </Description>
    </Property>
    <Property ovf:key="maxClients" ovf:type="uint16" ovf:value="256"
              ovf:userConfigurable="true">
         <Description>Limit the number of simultaneous requests that will be served.
</Description>
    </Property>
</ProductSection>
```

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

```
<Environment
    ...
    <!--- Properties defined for this virtual machine -->
    <PropertySection>
      <Property ovfenv:name="org.apache.httpd.httpPort ovfenv:value="80"/>
      <Property ovfenv:name="org.apache.httpd.httpsPort ovfenv:value="443"/>
      <Property ovfenv:name="org.apache.httpd.startThreads” ovfenv:value="50"/>
      <Property ovfenv:name="org.apache.httpd.minSpareThreads" ovfenv:value="15"/>
      <Property ovfenv:name="org.apache.httpd.maxSpareThreads" ovfenv:value="30"/>
      <Property ovfenv:name="org.apache.httpd.maxClients" ovfenv:value="256"/>
      ...
    </PropertySection>
    ...
</Environment>
```

Simple LAMP OVF Descriptor

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

```
<?xml version="1.0" encoding="UTF-8"?>
<Envelope
    xmlns="http://schemas.dmtf.org/ovf/envelope/1"
    xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
    xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
schema/2/CIM_ResourceAllocationSettingData"
    <!-- References to all external files -->
    <References>
        <File ovf:id="lamp" ovf:href="lamp.vmdk" ovf:size="180114671"/>
    </References>
    <!-- Describes meta-information about all virtual disks in the package.  -->
    <DiskSection>
        <Info>List of the virtual disks used in the package</Info>
        <Disk ovf:diskId="lamp" ovf:fileRef="lamp" ovf:capacity="4294967296"
            ovf:populatedSize="1924967692"
            ovf:format="http://www.vmware.com/specifications/vmdk.html#streamOptimized"/>
    </DiskSection>
    <!-- Describes all networks used in the package -->
    <NetworkSection>
```

```
        <Info>Logical networks used in the package</Info>
        <Network ovf:name="VM Network">
            <Description>The network that the LAMP Service will be available
            on</Description>
        </Network>
    </NetworkSection>
    <VirtualSystem ovf:id="MyLampService">
        <Info>Single-VM Virtual appliance with LAMP stack</Info>
        <Name>LAMP Virtual Appliance</Name>
        <!-- Overall information about the product -->
        <ProductSection>
            <Info>Product information for the service</Info>
            <Product>Lamp Service</Product>
            <Version>1.0</Version>
            <FullVersion>1.0.0</FullVersion>
        </ProductSection>
        <!-- Linux component configuration parameters -->
        <ProductSection ovf:class="org.linuxdistx">
            <Info>Product customization for the installed Linux system</Info>
            <Product>Linux Distribution X</Product>
            <Version>2.6.3</Version>
            <Property ovf:key="hostname" ovf:type="string">
                <Description>Specifies the hostname for the appliance</Description>
            </Property>
            <Property ovf:key="ip" ovf:type="string">
                <Description>Specifies the IP address for the appliance</Description>
            </Property>
            <Property ovf:key="subnet" ovf:type="string">
                <Description> Specifies the subnet to use on the deployed network
                </Description>
            </Property>
            <Property ovf:key="gateway" ovf:type="string">
                <Description> Specifies the gateway on the deployed network
                </Description>
            </Property>
            <Property ovf:key="dns" ovf:type="string">
                <Description> A comma separated list of DNS servers on the deployed
                    network </Description>
            </Property>
            <Property ovf:key="netCoreRmemMaxMB" ovf:type="uint16" ovf:value="16"
                ovf:userConfigurable="true">
                <Description> Specify TCP read max buffer size in mega bytes. Default is
                    16. </Description>
            </Property>
            <Property ovf:key="netCoreWmemMaxMB" ovf:type="uint16" ovf:value="16"
                ovf:userConfigurable="true">
                <Description> Specify TCP write max buffer size in mega bytes. Default is
                    16. </Description>
            </Property>
        </ProductSection>
        <!-- Apache  component configuration parameters -->
        <ProductSection ovf:class="org.apache.httpd">
            <Info>Product customization for the installed Apache Web Server</Info>
            <Product>Apache Distribution Y</Product>
            <Version>2.6.6</Version>
            <Property ovf:key="httpPort" ovf:type="uint16" ovf:value="80"
                ovf:userConfigurable="true">
                <Description>Port number for HTTP requests</Description>
            </Property>
            <Property ovf:key="httpsPort" ovf:type="uint16" ovf:value="443"
                ovf:userConfigurable="true">
                <Description>Port number for HTTPS requests</Description>
            </Property>
            <Property ovf:key="startThreads" ovf:type="uint16" ovf:value="50"
                ovf:userConfigurable="true">
                <Description>Number of threads created on startup. </Description>
            </Property>
            <Property ovf:key="minSpareThreads" ovf:type="uint16" ovf:value="15"
                ovf:userConfigurable="true">
                <Description> Minimum number of idle threads to handle request spikes.
                </Description>
            </Property>
            <Property ovf:key="maxSpareThreads" ovf:type="uint16" ovf:value="30"
                ovf:userConfigurable="true">
```

```
                <Description>Maximum number of idle threads </Description>
            </Property>
            <Property ovf:key="maxClients" ovf:type="uint16" ovf:value="256"
                ovf:userConfigurable="true">
                <Description>Limit the number of simultaneous requests that will be
                    served. </Description>
            </Property>
        </ProductSection>
        <!-- MySQL  component configuration parameters -->
        <ProductSection ovf:class="org.mysql.db">
            <Info>Product customization for the installed MySql Database Server</Info>
            <Product>MySQL Distribution Z</Product>
            <Version>5.0</Version>
            <Property ovf:key="queryCacheSizeMB" ovf:type="uint16" ovf:value="32"
                ovf:userConfigurable="true">
                <Description>Buffer to cache repeated queries for faster access (in
                MB)</Description>
            </Property>
            <Property ovf:key="maxConnections" ovf:type="uint16" ovf:value="500"
                ovf:userConfigurable="true">
                <Description>The number of concurrent connections that can be
                served</Description>
            </Property>
            <Property ovf:key="waitTimeout" ovf:type="uint16" ovf:value="100"
                ovf:userConfigurable="true">
                <Description>Number of seconds to wait before timing out a connection
                </Description>
            </Property>
        </ProductSection>
        <!-- PHP component configuration parameters -->
        <ProductSection ovf:class="net.php">
            <Info>Product customization for the installed PHP component</Info>
            <Product>PHP Distribution U</Product>
            <Version>5.0</Version>
            <Property ovf:key="sessionTimeout" ovf:type="uint16" ovf:value="5"
                ovf:userConfigurable="true">
                <Description> How many minutes a session has to be idle before it is
                    timed out </Description>
            </Property>
            <Property ovf:key="concurrentSessions" ovf:type="uint16" ovf:value="500"
                ovf:userConfigurable="true">
                <Description> The number of concurrent sessions that can be served
                </Description>
            </Property>
            <Property ovf:key="memoryLimit" ovf:type="uint16" ovf:value="32"
                ovf:userConfigurable="true">
                <Description> How much memory in megabytes a script can consume before
                    being killed </Description>
            </Property>
        </ProductSection>
        <OperatingSystemSection ovf:id="99">
            <Info>Guest Operating System</Info>
            <Description>Linux 2.6.x</Description>
        </OperatingSystemSection>
        <VirtualHardwareSection>
            <Info>Virtual Hardware Requirements: 256MB, 1 CPU, 1 disk, 1 NIC</Info>
            <System>
                <vssd:ElementName>Virtual Hardware Family</vssd:ElementName>
                <vssd:InstanceID>0</vssd:InstanceID>
                <vssd:VirtualSystemType>vmx-04</vssd:VirtualSystemType>
            </System>
            <Item>
                <rasd:Description>Number of virtual CPUs</rasd:Description>
                <rasd:ElementName>1 virtual CPU</rasd:ElementName>
                <rasd:InstanceID>1</rasd:InstanceID>
                <rasd:ResourceType>3</rasd:ResourceType>
                <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
            </Item>
            <Item>
                <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
                <rasd:Description>Memory Size</rasd:Description>
                <rasd:ElementName>256 MB of memory</rasd:ElementName>
                <rasd:InstanceID>2</rasd:InstanceID>
                <rasd:ResourceType>4</rasd:ResourceType>
```

```
                <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
            </Item>
            <Item>
                <rasd:AutomaticAllocation>true</rasd:AutomaticAllocation>
                <rasd:Connection>VM Network</rasd:Connection>
                <rasd:ElementName>Ethernet adapter on "VM Network"</rasd:ElementName>
                <rasd:InstanceID>3</rasd:InstanceID>
                <rasd:ResourceType>10</rasd:ResourceType>
            </Item>
            <Item>
                <rasd:ElementName>SCSI Controller 0 - LSI Logic</rasd:ElementName>
                <rasd:InstanceID>4</rasd:InstanceID>
                <rasd:ResourceSubType>LsiLogic</rasd:ResourceSubType>
                <rasd:ResourceType>6</rasd:ResourceType>
            </Item>
            <Item>
                <rasd:ElementName>Harddisk 1</rasd:ElementName>
                <rasd:HostResource>ovf:/disk/lamp</rasd:HostResource>
                <rasd:InstanceID>5</rasd:InstanceID>
                <rasd:Parent>4</rasd:Parent>
                <rasd:ResourceType>17</rasd:ResourceType>
            </Item>
        </VirtualHardwareSection>
    </VirtualSystem>
</Envelope>
```

Two-tier LAMP OVF Descriptor

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

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

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

- Alll the user-configurable parameters must be put in the *VirtualSystemCollection* entity*.* The ProductSections for Apache, MySQL, and PHP are unchanged from the single VM case.

- The Linux software in the two virtual machines needs to be configured slightly different (IP and hostname) while sharing most parameters. A new ProductSection is added to the *VirtualSystemCollection* to prompt the user, and the `${property}` expression is used to assign the values in each *VirtualSystem* entity.

- Disk chains are used to keep the download size comparable to that of a single VM appliance. Since the Linux installation is stored on a shared base disk, effectively only one copy of Linux needs to be downloaded.

The complete OVF descriptor is shown below:

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

```
         This example is encoded as a delta-disk hierarchy.
    -->
    <DiskSection>
        <Info>List of the virtual disks used in the package</Info>
        <Disk ovf:diskId="lamp-base" ovf:fileRef="lamp-base" ovf:capacity="4294967296"
            ovf:populatedSize="1924967692"
            ovf:format="http://www.vmware.com/specifications/vmdk.html#streamOptimized"/>
        <Disk ovf:diskId="lamp-db" ovf:fileRef="lamp-db" ovf:capacity="4294967296"
            ovf:populatedSize="19249672"
            ovf:format="http://www.vmware.com/specifications/vmdk.html#streamOptimized"
            ovf:parentRef="lamp-base"/>
        <Disk ovf:diskId="lamp-app" ovf:fileRef="lamp-app" ovf:capacity="4294967296"
            ovf:populatedSize="2349692"
            ovf:format="http://www.vmware.com/specifications/vmdk.html#streamOptimized"
            ovf:parentRef="lamp-base"/>
    </DiskSection>
    <!-- Describes all networks used in the package -->
    <NetworkSection>
        <Info>Logical networks used in the package</Info>
        <Network ovf:name="VM Network">
            <Description>The network that the LAMP Service will be available
            on</Description>
        </Network>
    </NetworkSection>
    <VirtualSystemCollection ovf:id="LampService">
        <Info>Virtual appliance with a 2-tier distributed LAMP stack</Info>
        <Name>LAMP Service</Name>
        <!-- Overall information about the product -->
        <ProductSection>
            <Info>Product information for the service</Info>
            <Product>My Lamp Service</Product>
            <Version>1.0</Version>
            <FullVersion>1.0.0</FullVersion>
        </ProductSection>
        <ProductSection>
            <Info>Product customization for Operating System Level</Info>
            <Product>Linux Distribution X</Product>
            <Version>2.6.3</Version>
            <Property ovf:key="dbHostname" ovf:type="string">
                <Description>Specifies the hostname for database virtual
                machine</Description>
            </Property>
            <Property ovf:key="appHostname" ovf:type="string">
                <Description>Specifies the hostname for application server virtual
                    machine</Description>
            </Property>
            <Property ovf:key="dbIp" ovf:type="string">
                <Description>Specifies the IP address for the database virtual
                machine</Description>
            </Property>
            <Property ovf:key="appIp" ovf:type="string">
                <Description>Specifies the IP address for application server
                VM</Description>
            </Property>
            <Property ovf:key="subnet" ovf:type="string">
                <Description> Specifies the subnet to use on the deployed network
                </Description>
            </Property>
            <Property ovf:key="gateway" ovf:type="string">
                <Description> Specifies the gateway on the deployed network
                </Description>
            </Property>
            <Property ovf:key="dns" ovf:type="string">
                <Description> A comma separated list of DNS servers on the deployed
                    network </Description>
            </Property>
            <Property ovf:key="netCoreRmemMaxMB" ovf:type="uint16" ovf:value="16"
                ovf:userConfigurable="true">
                <Description> Specify TCP read max buffer size in mega bytes. Default is
                    16. </Description>
            </Property>
            <Property ovf:key="netCoreWmemMaxMB" ovf:type="uint16" ovf:value="16"
                ovf:userConfigurable="true">
                <Description> Specify TCP write max buffer size in mega bytes. Default is
```

```
                    16. </Description>
            </Property>
        </ProductSection>
        <!-- Apache  component configuration parameters -->
        <ProductSection ovf:class="org.apache.httpd">
            <Info>Product customization for the installed Apache Web Server</Info>
            <Product>Apache Distribution Y</Product>
            <Version>2.6.6</Version>
            <Property ovf:key="httpPort" ovf:type="uint16" ovf:value="80"
                ovf:userConfigurable="true">
                <Description>Port number for HTTP requests</Description>
            </Property>
            <Property ovf:key="httpsPort" ovf:type="uint16" ovf:value="443"
                ovf:userConfigurable="true">
                <Description>Port number for HTTPS requests</Description>
            </Property>
            <Property ovf:key="startThreads" ovf:type="uint16" ovf:value="50"
                ovf:userConfigurable="true">
                <Description>Number of threads created on startup. </Description>
            </Property>
            <Property ovf:key="minSpareThreads" ovf:type="uint16" ovf:value="15"
                ovf:userConfigurable="true">
                <Description>Minimum number of idle threads to handle request spikes.
                </Description>
            </Property>
            <Property ovf:key="maxSpareThreads" ovf:type="uint16" ovf:value="30"
                ovf:userConfigurable="true">
                <Description>Maximum number of idle threads </Description>
            </Property>
            <Property ovf:key="maxClients" ovf:type="uint16" ovf:value="256"
                ovf:userConfigurable="true">
                <Description>Limits the number of simultaneous requests that will be
                    served. </Description>
            </Property>
        </ProductSection>
        <!-- MySQL  component configuration parameters -->
        <ProductSection ovf:class="org.mysql.db">
            <Info>Product customization for the installed MySql Database Server</Info>
            <Product>MySQL Distribution Z</Product>
            <Version>5.0</Version>
            <Property ovf:key="queryCacheSizeMB" ovf:type="uint16" ovf:value="32"
                ovf:userConfigurable="true">
                <Description>Buffer to cache repeated queries for faster access (in
                MB)</Description>
            </Property>
            <Property ovf:key="maxConnections" ovf:type="uint16" ovf:value="500"
                ovf:userConfigurable="true">
                <Description>The number of concurrent connections that can be
                served</Description>
            </Property>
            <Property ovf:key="waitTimeout" ovf:type="uint16" ovf:value="100"
                ovf:userConfigurable="true">
                <Description>Number of seconds to wait before timing out a connection
                </Description>
            </Property>
        </ProductSection>
        <!-- PHP component configuration parameters -->
        <ProductSection ovf:class="net.php">
            <Info>Product customization for the installed PHP component</Info>
            <Product>PHP Distribution U</Product>
            <Version>5.0</Version>
            <Property ovf:key="sessionTimeout" ovf:type="uint16" ovf:value="5"
                ovf:userConfigurable="true">
                <Description> How many minutes a session has to be idle before it is
                    timed out </Description>
            </Property>
            <Property ovf:key="concurrentSessions" ovf:type="uint16" ovf:value="500"
                ovf:userConfigurable="true">
                <Description> The number of concurrent sessions that can be served
                </Description>
            </Property>
            <Property ovf:key="memoryLimit" ovf:type="uint16" ovf:value="32"
                ovf:userConfigurable="true">
                <Description> How much memory in megabytes a script can consume before
```

```
                    being killed </Description>
            </Property>
        </ProductSection>
        <StartupSection>
            <Info>Startup order of the virtual machines</Info>
            <Item ovf:id="DbServer" ovf:order="1" ovf:startDelay="120"
                ovf:startAction="powerOn" ovf:waitingForGuest="true" ovf:stopDelay="120"
                ovf:stopAction="guestShutdown"/>
            <Item ovf:id="AppServer" ovf:order="2" ovf:startDelay="120"
                ovf:startAction="powerOn" ovf:waitingForGuest="true" ovf:stopDelay="120"
                ovf:stopAction="guestShutdown"/>
        </StartupSection>
        <VirtualSystem ovf:id="AppServer">
            <Info>The configuration of the AppServer virtual machine</Info>
            <Name>Application Server</Name>
            <!-- Linux component configuration parameters -->
            <ProductSection ovf:class="org.linuxdistx">
                <Info>Product customization for the installed Linux system</Info>
                <Product>Linux Distribution X</Product>
                <Version>2.6.3</Version>
                <Property ovf:key="hostname" ovf:type="string" ovf:value="${appHostName}"/>
                <Property ovf:key="ip" ovf:type="string" ovf:value="${appIp}"/>
                <Property ovf:key="subnet" ovf:type="string" ovf:value="${subnet}"/>
                <Property ovf:key="gateway" ovf:type="string" ovf:value="${gateway}"/>
                <Property ovf:key="dns" ovf:type="string" ovf:value="${dns}"/>
                <Property ovf:key="netCoreRmemMaxMB" ovf:type="string"
                    ovf:value="${netCoreRmemMaxMB}"/>
                <Property ovf:key="netCoreWmemMaxMB" ovf:type="string"
                    ovf:value="${netCoreWmemMaxMB}"/>
            </ProductSection>
            <OperatingSystemSection ovf:id="99">
                <Info>Guest Operating System</Info>
                <Description>Linux 2.6.x</Description>
            </OperatingSystemSection>
            <VirtualHardwareSection>
                <Info>Virtual Hardware Requirements: 256 MB, 1 CPU, 1 disk, 1 NIC</Info>
                <System>
                    <vssd:ElementName>Virtual Hardware Family</vssd:ElementName>
                    <vssd:InstanceID>0</vssd:InstanceID>
                    <vssd:VirtualSystemType>vmx-04</vssd:VirtualSystemType>
                </System>
                <Item>
                    <rasd:Description>Number of virtual CPUs</rasd:Description>
                    <rasd:ElementName>1 virtual CPU</rasd:ElementName>
                    <rasd:InstanceID>1</rasd:InstanceID>
                    <rasd:ResourceType>3</rasd:ResourceType>
                    <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
                </Item>
                <Item>
                    <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
                    <rasd:Description>Memory Size</rasd:Description>
                    <rasd:ElementName>256 MB of memory</rasd:ElementName>
                    <rasd:InstanceID>2</rasd:InstanceID>
                    <rasd:ResourceType>4</rasd:ResourceType>
                    <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
                </Item>
                <Item>
                    <rasd:AutomaticAllocation>true</rasd:AutomaticAllocation>
                    <rasd:Connection>VM Network</rasd:Connection>
                    <rasd:ElementName>Ethernet adapter on "VM Network"</rasd:ElementName>
                    <rasd:InstanceID>3</rasd:InstanceID>
                    <rasd:ResourceSubType>PCNet32</rasd:ResourceSubType>
                    <rasd:ResourceType>10</rasd:ResourceType>
                </Item>
                <Item>
                    <rasd:ElementName>SCSI Controller 0 - LSI Logic</rasd:ElementName>
                    <rasd:InstanceID>4</rasd:InstanceID>
                    <rasd:ResourceSubType>LsiLogic</rasd:ResourceSubType>
                    <rasd:ResourceType>6</rasd:ResourceType>
                </Item>
                <Item>
                    <rasd:ElementName>Harddisk 1</rasd:ElementName>
                    <rasd:HostResource>ovf:/disk/lamp-app</rasd:HostResource>
                    <rasd:InstanceID>5</rasd:InstanceID>
```

```
                    <rasd:Parent>4</rasd:Parent>
                    <rasd:ResourceType>17</rasd:ResourceType>
                </Item>
            </VirtualHardwareSection>
        </VirtualSystem>
        <VirtualSystem ovf:id="DB Server">
            <Info>The configuration of the database virtual machine</Info>
            <Name>Database Server</Name>
            <!-- Linux component configuration parameters -->
            <ProductSection ovf:class="org.linuxdistx">
                <Info>Product customization for the installed Linux system</Info>
                <Product>Linux Distribution X</Product>
                <Version>2.6.3</Version>
                <Property ovf:key="hostname" ovf:type="string"
                    ovf:value="${dbHostName}"/>
                <Property ovf:key="ip" ovf:type="string" ovf:value="${dbIp}"/>
                <Property ovf:key="subnet" ovf:type="string" ovf:value="${subnet}"/>
                <Property ovf:key="gateway" ovf:type="string" ovf:value="${gateway}"/>
                <Property ovf:key="dns" ovf:type="string" ovf:value="${dns}"/>
                <Property ovf:key="netCoreRmemMaxMB" ovf:type="string"
                    ovf:value="${netCoreRmemMaxMB}"/>
                <Property ovf:key="netCoreWmemMaxMB" ovf:type="string"
                    ovf:value="${netCoreWmemMaxMB}"/>
            </ProductSection>
            <OperatingSystemSection ovf:id="99">
                <Info>Guest Operating System</Info>
                <Description>Linux 2.6.x</Description>
            </OperatingSystemSection>
            <VirtualHardwareSection>
                <Info>Virtual Hardware Requirements: 256 MB, 1 CPU, 1 disk, 1 nic</Info>
                <System>
                    <vssd:ElementName>Virtual Hardware Family</vssd:ElementName>
                    <vssd:InstanceID>0</vssd:InstanceID>
                    <vssd:VirtualSystemType>vmx-04</vssd:VirtualSystemType>
                </System>
                <Item>
                    <rasd:Description>Number of virtual CPUs</rasd:Description>
                    <rasd:ElementName>1 virtual CPU</rasd:ElementName>
                    <rasd:InstanceID>1</rasd:InstanceID>
                    <rasd:ResourceType>3</rasd:ResourceType>
                    <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
                </Item>
                <Item>
                    <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
                    <rasd:Description>Memory Size</rasd:Description>
                    <rasd:ElementName>256 MB of memory</rasd:ElementName>
                    <rasd:InstanceID>2</rasd:InstanceID>
                    <rasd:ResourceType>4</rasd:ResourceType>
                    <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
                </Item>
                <Item>
                    <rasd:AutomaticAllocation>true</rasd:AutomaticAllocation>
                    <rasd:Connection>VM Network</rasd:Connection>
                    <rasd:ElementName>Ethernet adapter on "VM Network"</rasd:ElementName>
                    <rasd:InstanceID>3</rasd:InstanceID>
                    <rasd:ResourceType>10</rasd:ResourceType>
                </Item>
                <Item>
                    <rasd:ElementName>SCSI Controller 0 - LSI Logic</rasd:ElementName>
                    <rasd:InstanceID>4</rasd:InstanceID>
                    <rasd:ResourceSubType>LsiLogic</rasd:ResourceSubType>
                    <rasd:ResourceType>6</rasd:ResourceType>
                </Item>
                <Item>
                    <rasd:ElementName>Harddisk 1</rasd:ElementName>
                    <rasd:HostResource>ovf:/disk/lamp-db</rasd:HostResource>
                    <rasd:InstanceID>5</rasd:InstanceID>
                    <rasd:Parent>4</rasd:Parent>
                    <rasd:ResourceType>17</rasd:ResourceType>
                </Item>
            </VirtualHardwareSection>
        </VirtualSystem>
    </VirtualSystemCollection>
</Envelope>
```

C Extensibility Example

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

- New section elements may be defined as part of the `Section` substitution group, and used wherever the OVF schemas allow sections to be present.
- 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"`.
- The OVF schemas allow additional attributes on existing types.

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.

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.

Custom Schema

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.

```
<?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">
```

```
        <xs:sequence>
            <xs:element name="Data" type="xs:string"/>
        </xs:sequence>
        <xs:attribute ref="ovf:required"/>
        <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:complexType>

</xs:schema >
```

Descriptor with custom extensions

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.

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

```
<?xml version="1.0" encoding="UTF-8"?>
<Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
    xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-
schema/2/CIM_ResourceAllocationSettingData"
    xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1"
    xmlns="http://schemas.dmtf.org/ovf/envelope/1"
    xmlns:custom="http://schemas.customextension.org/1">

    <!-- Dummy References element -->
    <References/>

    <!-- EXAMPLE: Optional custom OVF section element with validation against custom schema -->
    <custom:CustomSection ovf:required="false">
        <Info>Description of custom extension</Info>
        <custom:Data>somevalue</custom:Data>
    </custom:CustomSection>

    <!-- Describes all networks used in the package -->
    <NetworkSection>
        <Info>Logical networks used in the package</Info>
        <!-- EXAMPLE: Optional custom attribute -->
        <Network ovf:name="VM Network" custom:desiredCapacity="1 Gbit/s"/>
        <!-- EXAMPLE: Optional custom meta-data inserted at extension point with validation
                     against custom schema -->
        <custom:CustomOther xsi:type="custom:CustomOther_Type" ovf:required="false">
            <custom:Data>somevalue</custom:Data>
        </custom:CustomOther>
    </NetworkSection>

    <!-- Dummy Content element -->
    <VirtualSystem ovf:id="Dummy">
        <Info>Dummy VirtualSystem</Info>
    </VirtualSystem>
</Envelope>
```

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