Document Identifier: DSP2017

Date: 2014-04-24

Version: 2.0.0

Open Virtualization Format White Paper

Document Type: DMTF Informational

Document Status: DMTF Informational

Document Language: en-US

Abstract

The Open Virtualization Format (OVF) White Paper describes the application of DSP0243, DSP8023, and DSP8027, the specifications that are part of the Open Virtualization Format (OVF) standard. The intended audience is anyone who wants to understand the OVF package and its application to specific use cases. Some familiarity with virtualization and the general concepts of the CIM model is assumed.

Contents

Figures

Tables

Foreword

The Open Virtualization Format White Paper (DSP2017) was prepared by the OVF Work Group of the DMTF.

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

Lawrence Lamers	VMware Inc. (Chair)
Marvin Waschke	DMTF Fellow (co-Editor)
Peter Wörndle	Ericsson AB (co-Editor)
Eric Wells	Hitachi, Ltd. (co-Editor)
Hemal Shah	Broadcom Corporation
Shishir Pardikar	Citrix Systems Inc.
Richard Landau	DMTF Fellow
Robert Freund	Hitachi, Ltd.
Jeff Wheeler	Huawei
Monica Martin	Microsoft Corporation
Cheng Wei	Microsoft Corporation
Srinivas Maturi	Oracle
Steffen Grarup	VMware Inc.
Rene Schmidt	VMware Inc.
Ghazanfar Ali	ZTE Corporation

Open Virtualization Format White Paper

1 Introduction

1.1 Overview

The Open Virtualization Format specification (OVF) provides the industry with a standard packaging format for software solutions based on virtual systems, solving critical business needs for software vendors and cloud computing service providers.

An OVF package can be used by an independent software vendor (ISV) to publish a software solution; by a data center operator to transport a software solution from one data center to another; by a customer to archive a software solution; or any other use case that can be met by having a standardized package for a software solution.

The following use cases are the main basis for OVF work:

1) the ability for ISVs to package a software solution that is capable of being used on more than one hypervisor

2) the ability to package a virtual system or collection of virtual systems so they can be moved from one data center to another

Other use cases and derivative use cases (i.e., subsets) are also applicable.

OVF version 1 has been widely adopted by the industry and is now an international standard.

OVF version 2 adds enhanced packaging capabilities, making it applicable to the broader range of use cases that are emerging as industry enters the cloud computing era.

OVF 2 adds the following features:

- Support for Network Ports
- Scaling at deployment time
- Support for basic placement policies
- Encryption of OVF packages
- Disk sharing at runtime
- Advanced Device Boot Order
- Advanced Data Transfer to Guest Software
- Support for Improved Internationalization - I18N
- Support of HASH Improved
- Updated CIM schema

OVF has adopted the Common Information Model (CIM), where appropriate, to allow management software to clearly understand and easily map resource properties by using an open standard. The `CIM_ResourceAllocationSettingData` class and its subclasses for specific device types is used to specify the resources needed for the virtual system to operate.

OVF 2 supports network configuration and the IEEE Edge Virtual Bridging Discovery and Configuration protocols that use *Network Port Profile* (DSP8049). The CIM_EthernetPortAllocationSettingData class provides the essential properties.

This document aims to give details of the motivation, goals, design and expected usage of OVF, and should be read in accompaniment with the OVF specification of the same major revision.

1.2 Design considerations

The rapid adoption of virtual infrastructure has highlighted the need for a standard, portable metadata format for the distribution of virtual systems onto and between virtualization platforms. The ability to package a software application together with the operating system on which it is certified, into a format 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 with the configuration of the virtual systems required to run them are called virtual appliances. In order to make this concept practical on a broad scale, it is important that the industry adopts a vendor-neutral standard for packaging such virtual appliances and the metadata required to automatically and securely install, configure, and run them on any virtualization platform.

From the user's point of view, OVF is a packaging format for virtual appliances. Once installed, an OVF package adds to the user’s infrastructure a self-contained, self-consistent, software application that provides a particular service or services. For example, an OVF package might contain a fully functional and tested web-server, database, and 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.

Whereas many virtual appliances contain only a single virtual system, modern enterprise applications are modeled as service oriented architectures (SOA) with multiple tiers, each containing one or more virtual systems. A single virtual system 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 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 OVF specification describes a hypervisor-neutral, efficient, extensible, and open format for the packaging and distribution of virtual appliances composed of one or more virtual systems. It aims to facilitate the automated and secure management not only of individual virtual systems, but also of the virtual appliance as a functional unit.

To be successful, OVF has been developed and endorsed by ISVs, virtual appliance vendors, operating system vendors, as well as virtual platform vendors. The OVF specification promotes customer confidence through the collaborative development of common standards for portability and interchange of virtual systems between different vendors’ virtualization platforms.

OVF 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 format for packaging virtual appliances.

The OVF specification is complimentary to existing IT management standards and frameworks and promotes best-of-breed competition through openness and extensibility. The explicit copyright notice attached to this document is intended to avoid arbitrary, independent, piecewise extensions to the format while permitting free distribution and implementation of the specification.

2 OVF key concepts

2.1 Virtual appliances

A virtual appliance is a pre-configured software stack comprising one or more virtual systems. Each virtual system is an independently installable run-time entity consisting of an operating system, applications and application-specific data, as well as metadata describing the virtual hardware that is required by the virtual system. 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 web, application and database tiers, 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 are changing the software distribution paradigm because they allow optimization of the software stack for the specific application and to deliver a turnkey service to the end user. For solution providers, building a virtual appliance is simpler and more cost effective than building a hardware appliance. The application is pre-packaged with the operating system that it uses, reducing compatibility testing and certification, allowing the software to be pre-installed in the environment in which it runs – by the ISV that develops the solution. For end users, virtual appliances offer an opportunity to dramatically simplify the software management lifecycle through the adoption of standardized, automated, and efficient processes that replace the individual OS and application specific management tasks used previously.

Virtual appliances need not be developed and delivered by third-party ISVs – the concept is equally useful and often used within an enterprise in which a virtual system 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 systems and communicating across the network by using a SOA model. Services are often composed of other services, which themselves might also be multi-tier applications, or composed of other services. Indeed the SOA model naturally fits into a virtual appliance-based infrastructure, because 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; an application server tier that implements the business logic; and a back-end database tier. A straightforward implementation divides this configuration into three virtual systems, one for each tier. In this way, the application can scale from a fraction of a single physical host to three physical hosts. Another approach is to treat each tier as a service in itself. Hence, each tier can scale to a multi-virtual system service that provides a clustered solution. Again, taking the web application as an example, a common scenario is to have many web servers, fewer applications servers, and one or two database servers. Implemented as virtual systems, each tier can scale across as many or as few physical machines as required, and each tier can support multiple instances of virtual systems that service requests.

One OVF may contain a single or many virtual systems. It is left to developers to decide which arrangement best suits their application. OVFs must be installed before they can be run; a particular virtualization platform may run the virtual system from the OVF, but this configuration is not required. If this configuration is chosen, the OVF itself can no longer be viewed as a “golden image” version of the appliance because run-time state for the virtual system(s) pervades the OVF. Moreover the digital signature that allows the platform to check the integrity of the OVF is 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 virtual system image formats, operating at the scope of a single virtual disk. Though they are frequently used as transport formats today, they are not designed to solve portability problems. They do not help with a virtual system that has multiple disks, or multiple virtual systems, or they need customization of the virtual system during installation. These formats provide no help if the virtual system is intended to run on multiple virtualization platforms (even if the virtualization platforms claim support of the particular virtual hard disk format used).

Within the OVF remit is the concept of the certification and integrity of a packaged virtual appliance. This concept allows the platform to determine the provenance of the appliance and permits 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 the appliance means that OVF packages may be implemented by using any operating system, and installed on any virtualization platform that supports the OVF format. A specific mechanism is provided for appliances to detect and react to the platform on which they are installed. This mechanism 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, multitier services and their associated distribution, installation, configuration, and execution:

- Provides direct support for the configuration of multi-tier applications and the composition of virtual systems to deliver composed services.
- Permits the specification of both virtual system and application-level configuration.
- Has 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.
- Uses commercially accepted procedures for integrity checking of the OVF contents, through the use of signatures and trusted third parties. This serves to reassure the consumer that an appliance has not been modified since it signed by the creator of the appliance. This assurance is seen as critical to the success of the virtual appliance market, and to the viability of independent creation and online download of appliances.
- Allows 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.

2.2 Life cycle

The life cycle for a virtual system is illustrated in Figure 1:

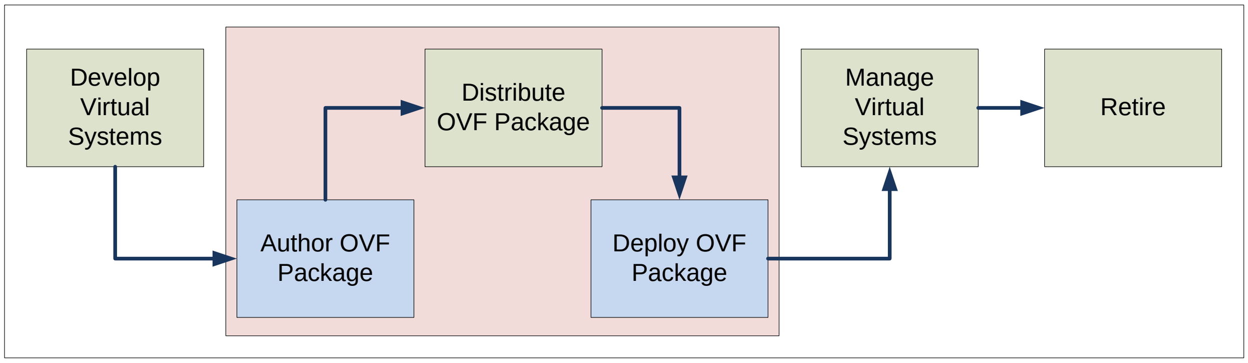


Figure 1 – OVF package life cycle

An OVF package is built from components that have been developed or acquired by the OVF author. These are packaged into a set of files that comprise a virtual appliance, consisting of one or more virtual machines and virtual machine collections and the relevant configuration and deployment metadata. For example, a clustered database component might be acquired from a third-party ISV. The installed service is then managed and eventually retired. Distribution, management, and retirement are outside the scope of OVF and are specific to the virtualization product used and the virtual appliance installed from the OVF. Management includes ongoing maintenance, configuration, and upgrade of the appliance. These activities depend on the installed service and environment, not the OVF package. The OVF specification focuses specifically on the authoring and deployment phases.

The OVF author function is illustrated in Figure 2.

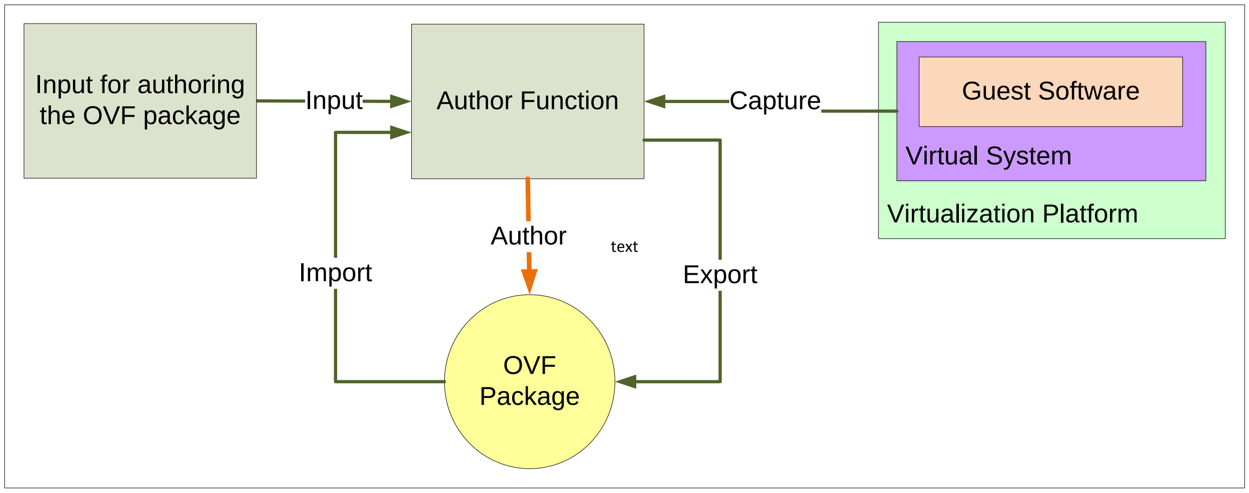


Figure 2 – OVF author function

An OVF package is authored in one of two ways. The straightforward method is to use a text editor or an XML authoring tool to create an OVF descriptor, assemble the required disk images and other files, and then create a tar file or file system that contains the OVF package.

An alternative method is to export an OVF package from a virtualization platform. The OVF descriptor may then be edited to include additional information. This method may be chosen for various reasons, including improving portability between virtualization platforms or providing options for configuration.

The OVF deployment function is illustrated in Figure 3 This diagram is also instructive as to the scope of work that is covered by the OVF work group.

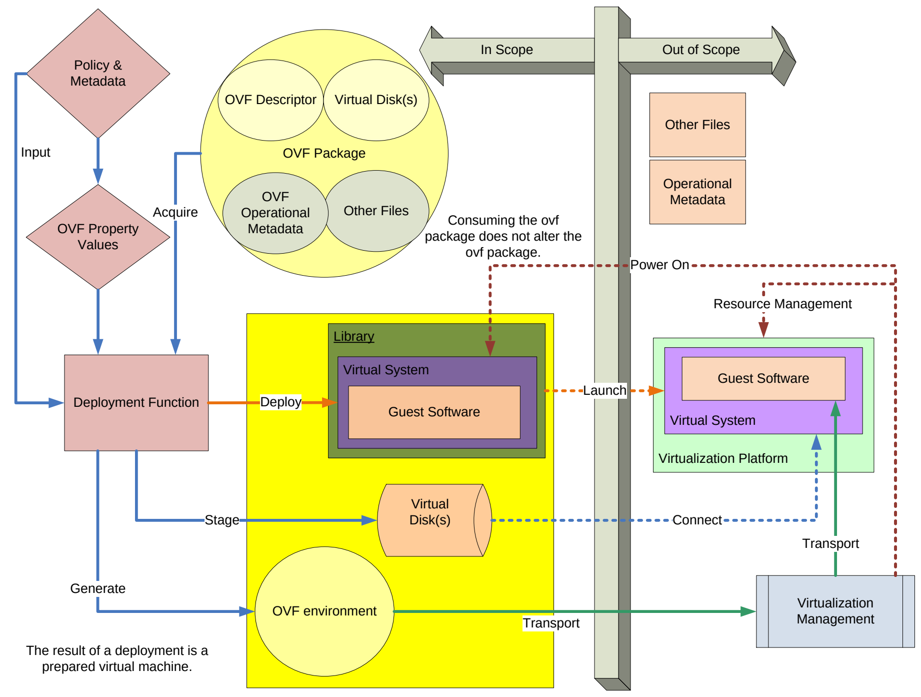


Figure 3 – OVF deployment function

The OVF operational metadata is information that may be needed for the proper operation of the virtual system or collection of virtual systems. The OVF operational metadata is a subset of the operational metadata that may be available when the virtual system is powered on.

As shown in the diagram, the OVF deployment function transports the OVF environment to the virtualization platform. The OVF specification is flexible on the exact nature of the transport, but it can be thought of as placing media, like a CD ROM, into a virtual reader on the virtual machine and the media being read by the guest operating system each time the system starts up. The metadata in the OVF environment is used for configuration after operating system startup and for meeting other requirements of guest software and virtualization platform for the proper operation of the virtual appliance.

3 XML backgrounder

3.1 Use of XML Schema

The OVF standard makes use of XML and XML Schema Definition Language (XSD or XSDL). XML is a markup language that defines a grammar for expressing XML elements and attributes, but says little about the structure of the elements and attributes in documents or the permissible data types for element and attribute values. Document Type Definitions (DTDs) have been part of XML from its inception and were intended to add structure and data types to XML, but they were found to be inadequate in many situations. In response, the W3C developed XSD, written itself in XML, which is a rich language for specifying XML document structure and data types. XSD and DTD can be used together, but XSD has become more common. XSD files are customarily given an ".xsd" extension.

An XSD file is metadata that describes the structure and data types of elements and data structures that are allowed in an XML document. XSD also supports inheritance and other object-oriented constructs.

These constructs simplify the creation of elaborate and complex XSDs, making them easier to write, more compact, and have fewer errors. Checking an XML document against the XML grammar only tests the document's syntax, but not the correctness of the structure and data in the document. Validating an XML document against an XSD checks the structure and data. When an XML document is consumed by another process, documents that are valid against an XSD are more likely to be processed without errors. In addition, some tools can generate XML document processors from XSDs. These tools can greatly accelerate development.

OVF uses XSD to specify the structure and data types of OVF descriptors, the XML documents in an OVF package that describe arbitrarily complex patterns for the instantiation of virtual systems. OVF descriptors have the file extension ".ovf". The official XSD for OVF 2.0.0 is found at http://schemas.dmtf.org/ovf/envelope/2/dsp8023.xsd. Users of OVF should use an XSD validation tool to check their OVF descriptors against the OVF XSD schema. A successfully validated descriptor is a necessary, but not sufficient, condition for OVF standard conformance. Some constraints in the standard specification go beyond the dsp8023.xsd and these must be checked manually for full conformance.

3.2 General XML concepts used in OVF

3.2.1 Element

An XML element is a data container in the OVF descriptor. Each element that is not empty begins with a start tag, followed by the elements contents, and the element is then closed with an end tag. The start tag consists of the element name, optionally followed by element attributes and surrounded by angled brackets, i.e., <elementname> or <elementname attributes>. The end tag consists of only the element name preceded by a / and enclosed in angle brackets, i.e., </elementname>. An element can contain other elements, text, attributes, or a combination of all of these.

XML elements follow these naming rules:

- Names can contain letters, numbers, and other characters.
- Names cannot start with a number or punctuation character.
- Names cannot start with the letters xml (or XML, or Xml, etc.).
- Names cannot contain spaces.
- Any name can be used; no words are reserved.

3.2.2 Attribute

XML elements can have attributes that provide additional information about an element. Attributes often provide information that is not a part of the data. Attribute values are always quoted by using single or double quotation marks.

3.2.3 Substitution group

A substitution group is an object-oriented feature of XSD that allows you to specify elements that can replace another element in documents generated from that schema. The replaceable element is called the head element and is defined in the schema's global scope. The elements of the substitution group are of the same type as the head element or a type that is derived from the head element's type.

In essence, a substitution group allows you to build a collection of elements that can substitute for a generic element. For example, if you are building an ordering system for a company that sells three types of widgets, you might define a generic widget element that contains common data for all three widget types. Then you can define a substitution group for the generic widget that contains more specific widget types that are derived from the generic widget. In the schema, an order can be defined as a sequence of generic widgets. In an XML file conformant with the schema, an order can be a sequence of widgets from the substitution group rather than generic widgets. Often the head element for a substitution group is declared to be abstract, so generic widgets cannot appear in the XML file.

4 OVF package

The OVF package provides a means to distribute software solutions deployed in a virtual system or collection of virtual systems. The OVF package consists of an OVF descriptor and related virtual disks. The OVF package exists as either a set of files referenced by a URL or a compressed file with the '.ova' extension.

4.1 Structure of an OVF descriptor

An OVF descriptor is a XML file. The root element of an OVF descriptor is the Envelope. The two most important child elements of an `Envelope` element are the `VirtualSystem` and `VirtualSystemCollection` elements. The `Envelope` element also contains sections that apply to all in `VirtualSystem` and `VirtualSystemCollection` elements the package. The `Envelope` element may contain both `VirtualSystem` and `VirtualSystemCollection` elements. The `VirtualSystemCollection` elements are a recursive construct that may, like the `Envelope` element, contain both `VirtualSystem` and `VirtualSystemCollection` elements. The OVF Schema defines a number of different sections. Some of these sections may only appear in the `Envelope` element. Others may only appear in the `VirtualSystem` element. Yet others may only appear in `VirtualSystemCollection element`. Some sections may appear in both `VirtualSystem` and `VirtualSystemCollection elements`. This structure is summarized in Figure 4.

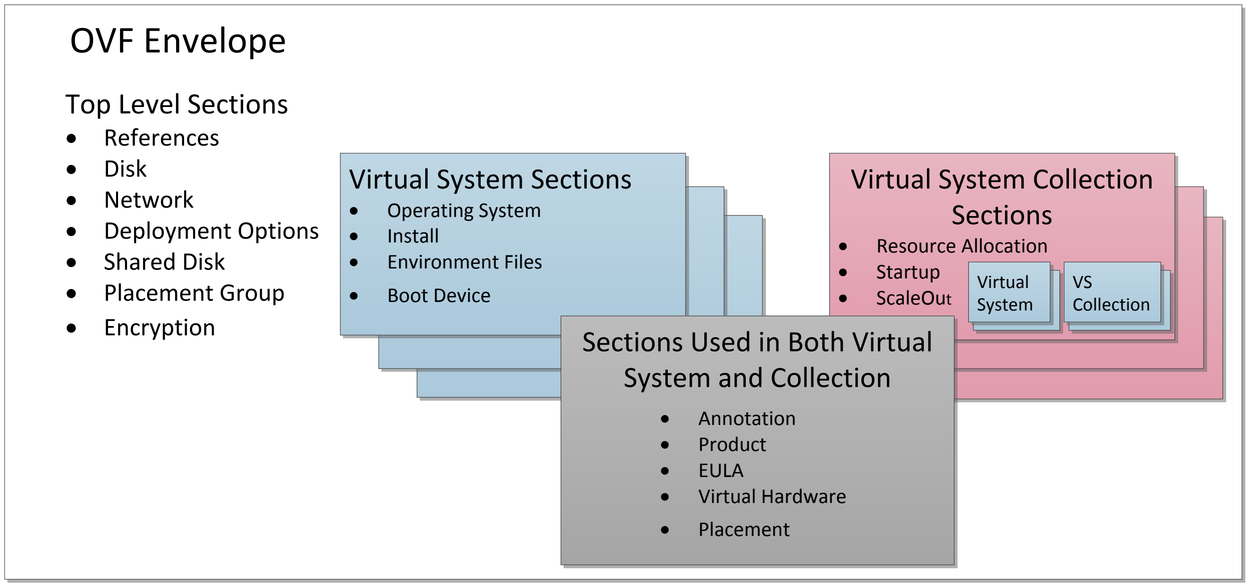


Figure 4 – OVF package structure

4.2 Global attributes defined in OVF Schema

The following OVF attributes defined in the OVF Schema are global attributes. OVF element specific OVF attributes are also defined. (See 4.4.and 4.3.)

- `required` attribute – indicates that the deployment should fail if the element is not present or is not understood. This attribute is an XSD Boolean with allowed values are 'true', 'false', '0', '1'. If not specified, the value is 'true' or '1'. This OVF attribute should not be confused with the XSD value `required` for the XSD `use` attribute. The two terms are similar but different in significance.
- `transport` attribute – is a space-separated list of supported transport types used to convey the information to the guest software. See 4.7.3 and 6.2.

- `configuration` attribute – identifies a configuration defined in a `Configuration` element in the `DeploymentOptionSection` element. See 4.4.6. The `configuration` attribute is used in the following places:
    - in the `VirtualHardwareSection` elements for `Item`, `EthernetPortItem`, `StorageItem` elements
    - in the `ResourceAllocationSection` element for `Item`, `EthernetPortItem`, `StorageItem` elements
    - in the `ScaleOutSection` elements for `InstanceCount` elements
    - in the `ProductSection` elements for `Property` elements
- `bound` attribute – is a range marker entry used to indicate minimum, normal, and maximum values for resource allocation setting data. The allowed values are 'min', 'normal', and 'max'. See 4.6.1 The `bound` attribute is used in two places to set limits on resource allocation:
    - in the `VirtualHardwareSection` elements for `Item`, `EthernetPortItem`, `StorageItem` elements
    - in the `ResourceAllocationSections` element for `Item`, `EthernetPortItem`, `StorageItem` elements

4.3 Extensibility in OVF

The OVF Schema use the XSD elements '`any`' and '`anyAttribute`' to extend the OVF descriptor and the OVF environment to provide custom metadata. This feature allows OVF packages to meet a wide variety of use cases in the industry.

The following definitions come from the XML schema reference website. See http://www.w3schools.com/schema/schema_elements_ref.asp.

- `any` – This definition enables the author to extend the XML document with elements not specified by the schema.
- `anyAttribute` – This definition enables the author to extend the XML document with attributes not specified by the schema.
- `##any` - Elements from any namespace are allowed. (This definition is the default.)
- `##other` - Elements from any namespace that is not the namespace of the parent element can be present.

An extension at the `Envelope` element level is done by defining a new member of the ovf:Section substitution group. An extension at the `Content` element level is done by defining a new member of the ovf:Section substitution group. See 3.2.3. These new section elements can be used where sections are allowed to be present by the OVF Schema. The `Info` element in each new section element can be used to give meaningful warnings to users when a new section element is being skipped because it is not understood by the deployment platform.

A type defined in the OVF Schema may be extended at the end with additional elements. Extension points are declared with an `xs:any` with a `namespace="##other"`.

Additional attributes are allowed in the OVF Schema. Extension points are declared with an `xs:anyAttribute.`

The `ovf:required` attribute specifies whether the information in the element is required or optional. The `ovf:required` attribute defaults to TRUE. If the deployment platform detects an element extension that is required and that it does not understand, it fails the deployment.

On custom attributes, the information in the attribute is not required for correct behavior.

```
EXAMPLE 1:
    <!—- Optional custom section example -->
    <otherns:IncidentTrackingSection ovf:required="false">
        <Info>Specifies information useful for incident tracking purposes</Info>
        <BuildSystem>Acme Corporation Official Build System</BuildSystem>
        <BuildNumber>102876</BuildNumber>
        <BuildDate>10-10-2008</BuildDate>
    </otherns:IncidentTrackingSection>

EXAMPLE 2:
    <!—- Open content example (extension of existing type) -->
    <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 -->
        <otherns:Author ovf:required="false">John Smith</otherns:Author>
    </AnnotationSection>

EXAMPLE 3:
    <!—- Optional custom attribute example -->
    <Network ovf:name="VM network" otherns:desiredCapacity="1 Gbit/s">
        <Description>The main network for VMs</Description>
    </Network>
```

4.4 OVF top level elements

The root element defined by the OVF Schema is the `Envelope` element. OVF elements that are direct children of an `Envelope` element are listed below in the order that they occur in the OVF descriptor:

- References element
- Section elements - a substitution group for section elements
- Content elements - a substitution group for content elements
- Strings element

Elements that are a substitution group for a `Section` element that is a direct child of the `Envelope` element:

- `DiskSection` element
- `NetworkSection` element
- `DeploymentOptionSection` element
- `SharedDiskSection` element
- `PlacementGroupSection` element
- `EncryptionSection` element

Elements that are a substitution group for a `Content` element:

- `VirtualSystem` element
- `VirtualSystemCollection` element (may be nested)

Elements that are a substitution group for a Section element used in a `Content` element are listed below in the order that they occur in an OVF descriptor:

- `AnnotationSection` element
- `ProductSection` element

- `OperatingSystemSection` element
- `EulaSection` element
- `VirtualHardwareSection` element
- `ResourceAllocationSection` element
- `InstallSection` element
- `StartupSection` element
- `EnvironmentFilesSection` element
- `BootDeviceSection` element
- `ScaleOutSection` element
- `PlacementSection` element

The additional elements defined in the OVF Schema are listed below. Note that these elements can be used in a namespace other than `ovf:`.

- Annotation
- AppUrl
- bootc:CIM_BootConfigSetting
- Category
- Configuration
- Content
- Description
- Disk
- EthernetPortItem
- File
- FullVersion
- Icon
- Info
- InstanceCount
- Item
- Label
- License
- Msg
- Name
- Network
- NetworkPortProfile
- NetworkPortProfileURI
- Product
- ProductUrl
- Property
- SharedDisk
- StorageItem
- System
- Value
- Vendor
- VendorUrl
- Version
- xenc:EncryptedKey
- xenc11:DerivedKey

An example of the basic structure of an OVF descriptor shown in Figure 4 is illustrated below.

```
ovf:Envelope
<xs:element name="References" type="ovf:References_Type">

```

```
<xs:element ref="ovf:Section" minOccurs="0" maxOccurs="unbounded">
<DiskSection>
  <Info> Describes all virtual disks used with the package </Info>
<NetworkSection>
  <Info>List of logical networks used in the package</Info>
<DeploymentOptionSection>
  <Info>List of deployment options available in the package</Info>

<xs:element ref="ovf:Content">
  <VirtualSystemCollection ovf:id="Acme VSC">
    <Info>The packaging of the first virtual appliance</Info>
------
    <VirtualSystem ovf:id="Acme VS 1">
     <Info>The packaging of the virtual machine 1</Info>
    <VirtualSystem ovf:id="Acme VS 2">
      <Info>The packaging of the virtual machine 2</Info>
------
    <VirtualSystemCollection ovf:id="Widget VSC">
           <Info>The packaging of the second virtual appliance</Info>
      <VirtualSystem ovf:id="Acme VS 3">
             <Info>The packaging of the virtual machine 3</Info>
      <VirtualSystem ovf:id="Acme VS 3">
        <Info>The packaging of the virtual machine 4</Info>
------
<xs:element name="Strings" type="ovf:Strings_Type" minOccurs="0"
maxOccurs="unbounded">
  <Info> Root element of I18N string bundle</Info>
```

4.4.1 VirtualSystem element

A `VirtualSystem` element is a substitution element for a `Content` element. It contains a number of `Section` elements that define a single virtual system. These section elements describe virtual hardware, the resource allocation, and product information applicable to the virtual system.

4.4.2 VirtualSystemCollection element

A `VirtualSystemCollection` element is a substitution element for a `Content` element. It contains one or more `VirtualSystem` elements and a number of `Section` elements that define a collection of virtual systems. These section elements describe virtual hardware, the resource allocation, and product information applicable to the virtual system collection.

4.4.3 References element

The `References` element contains the references to all external files.

4.4.4 DiskSection element

The `DiskSection` element defines the virtual disks used by the virtual systems in the OVF package.

Any virtual disk format may be used, as long as the virtual disk format specification is public and available 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 may be a simple, basic disk block format agnostic to the guest software installed. For 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 run-time virtual disk format may be identical to the distribution format, but is often different because it may not be efficient to run out of a compressed virtual disk format. The guest software that is installed has its own file system format, e.g., NTFS, EXT3, or ZFS. The OVF virtual disk though does not need to know the file system format.

The following example shows a description of virtual disks:

```
<DiskSection>
    <Info>Describes the set of virtual disks</Info>
    <Disk ovf:diskId="vmdisk1" ovf:fileRef="file1" ovf:capacity="8589934592"
          ovf:populatedSize="3549324972"
          ovf:format=
              "http://www.vmware.com/interfaces/specifications/vmdk.html#sparse">
    </Disk>
    <Disk ovf:diskId="vmdisk2" ovf:capacity="536870912"
    </Disk>
    <Disk ovf:diskId="vmdisk3" ovf:capacity="${disk.size}"
          ovf:capacityAllocationUnits="byte * 2^30"
    </Disk>
</DiskSection>
```

4.4.5 NetworkSection element

The `NetworkSection` element describes the network and network connections used when the OVF package is deployed. The network may be defined in simplistic terms, e.g., as a Red, Green, Blue network. The Ethernet Port characteristics are defined by the `CIM_EthernetPortAllocationSettingData` class. It may also be defined in greater detail with the use of the *Network Port Profile* (DSP8049), which may be included in an OVF package.

There is a basic assumption that a flat layer 2 network is available for which the Ethernet ports to connect. There is no assumption made regarding the services or characteristics of that network. The speed of the Ethernet port can be set by using the CIM_EthernetPortAllocationSettingData.

4.4.5.1 OVF networks

The following example is a snippet of an OVF descriptor for the Network Section. It simply states that the Ethernet ports defined in the virtual systems are attached to the “Red” network. The `epasd:Connection` property specifies the network to which the Ethernet port is connected. The simplistic assumption is that all Ethernet ports are connected to a layer 2 network. If more than one network is defined, e.g., “Red” and “Green”, there is an assumption that the “Red” and “Green” networks are not connected. A connection to an external network, e.g., a data center LAN or a WAN, is considered part of the deployment provisioning and not specified in the OVF descriptor. In this case the agreement between the consumer of the OVF package and the hosting service determines the network characteristics and services.

```
    <!-- Describes all networks used in the package -->
    <NetworkSection>
        <Info>Logical networks used in the package</Info>
        <Network ovf:name="Red Network">
            <Description>The network that the virtual systems are attached to.
            on</Description>
        </Network>
    </NetworkSection>

```

The following example is a snippet of an item in an OVF descriptor `VirtualHardwareSection` in a `VirtualSystem` element that illustrates how the virtual system is attached to the “Red” network by using the `Connection` property from the `CIM_EthernetPortAllocationSettingData` class.

```
<EthernetPortItem>
  <epasd:AddressOnParent>7</epasd:AddressOnParent>
  <epasd:AutomaticAllocation>true</epasd:AutomaticAllocation>
  <epasd:Connection>Red Network</epasd:Connection>
  <epasd:Description>Virtual Ethernet adapter</epasd:Description>
  <epasd:ElementName>Virtual NIC 1</epasd:ElementName>
  <epasd:InstanceID>8</epasd:InstanceID>
  <epasd:ResourceType>10</epasd:ResourceType>
</EthernetPortItem>
```

More than one network may be defined.

Figure 5 illustrates a simplistic network. The Ethernet port connects to the “Red” network.

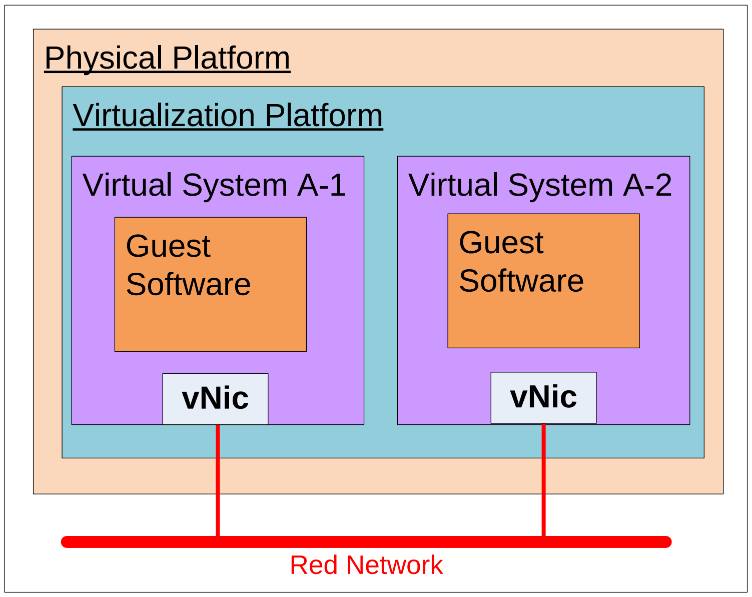


Figure 5 – Network connections

Figure 6 illustrates a dual network configuration: a “Green” network for storage connections and a “Red” network for local area network connections.

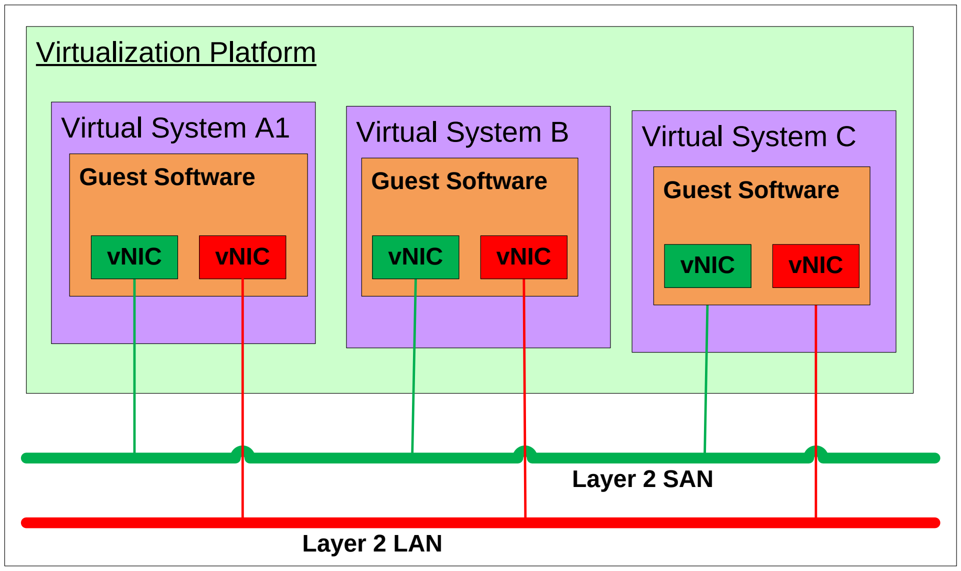


Figure 6 – LAN-SAN network connections

4.4.5.2 Network Port Profile

The *Network Port Profile* (DSP8049) defines a configuration of the properties of a network's network ports that is used for communication by the virtual systems defined in the OVF package. See the *Virtual Networking Management White Paper* (DSP2025) for additional information.

A complete listing of the properties in the `EthernetPortAllocationSettingData` class that can be specified is shown below. The allowed values for each property are defined in the DSP0243, DSP1041, DSP1050, and the CIM Schema. Only a subset of these properties is used in most OVF descriptors.

```
<EthernetPortItem>
  <epasd:Address>
  <epasd:AddressOnParent>
  <epasd:AllocationUnits>
  <epasd:AllowedPriorities>
  <epasd:AllowedToReceiveMACAddresses>
  <epasd:AllowedToReceiveVLANs>
  <epasd:AllowedToTransmitMACAddresses>
  <epasd:AllowedToTransmitVLANs>
  <epasd:AutomaticAllocation>
  <epasd:AutomaticDeallocation>
  <epasd:Caption>
  <epasd:ChangeableType>
  <epasd:ConfigurationName>
  <epasd:Connection>
  <epasd:ConsumerVisibility>
  <epasd:DefaultPortVID>
  <epasd:DefaultPriority>
  <epasd:Description>
  <epasd:DesiredVLANEndpointMode>
  <epasd:ElementName>
  <epasd:GroupID>
  <epasd:HostResource>
  <epasd:InstanceID>
  <epasd:Limit>
  <epasd:ManagerID>
  <epasd:MappingBehavior>
  <epasd:NetworkPortProfileID>
  <epasd:NetworkPortProfileIDType>
  <epasd:OtherEndpointMode>
  <epasd:OtherNetworkPortProfileIDTypeInfo>
  <epasd:OtherResourceType>
  <epasd:Parent>
  <epasd:PoolID>
  <epasd:PortCorrelationID>
  <epasd:PortVID>
  <epasd:Promiscuous>
  <epasd:ReceiveBandwidthLimit>
  <epasd:ReceiveBandwidthReservation>
  <epasd:Reservation>
  <epasd:ResourceSubType>
  <epasd:ResourceType>
  <epasd:SourceMACFilteringEnabled>
  <epasd:VSITypeID>
  <epasd:VSITypeIDVersion>
  <epasd:VirtualQuantity>
  <epasd:VirtualQuantityUnits>
  <epasd:Weight>
</EthernetPortItem>
```

4.4.6 DeploymentOptionsSection element

The `DeploymentOptionsSection` element is a direct child element of the `Envelope` element. The `Configuration` element is a direct child element of the `DeploymentOptionsSection` element. A `DeploymentOptionsSection` element contains one or more `Configuration` elements.

The `Configuration` element is used to specify a resource configuration used when an OVF package is deployed. A choice of one of the configurations is made at deployment. The user input can be requested through the use of the `userConfigurable` attribute.

The `DeploymentOptionsSection` element lists the IDs, labels and descriptions of the configurations available in the OVF package.

A default configuration is indicated by setting the `default` attribute to `true`. In the absence of other input, the default configuration is used. If a default is not indicated, the first configuration is taken as the default.

Each configuration has a unique `ID` attribute that identifies that configuration. This value of the `ID` attribute is specified in a `configuration` attribute in other section elements, such as a `VirtualHardwareSection` element, a `ProductSection` element, or a `ScaleOutSection` element.

If an element is to appear in more than one configuration, the `configuration` attribute of the elements is a list of space-separated configuration IDs.

The deployment function needs to select a configuration either from user input or from other metadata. When a configuration is selected, elements with a `configuration` attribute set to the selected configuration are used for the deployment. Elements without a `configuration` attribute are also deployed, but those with a different configuration attribute are not deployed. Thus, selecting a single configuration during deployment may affect the configuration of many different items in different sections of the OVF package.

A snippet from an OVF descriptor `DeploymentOptionsSection` element that illustrates the use of the `Configuration` element is shown below. Note that the default configuration for the memory resource allocation is highlighted in blue and the "big" configuration is highlighted in gray. The deployment function chooses the default configuration unless there is consumer input to choose the "big" configuration.

```
<DeploymentOptionsSection>
   <Configuration ovf:id="big">
      <Label>Big</Label>
      <Description>Apply reservations for Memory</Description>
   </Configuration>
   ...
</DeploymentOptionsSection>

<VirtualHardwareSection>
  <Info>...</Info>

  <Item>
    <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
    <rasd:ElementName>512 MB memory size-no reservation</rasd:ElementName>
    <rasd:InstanceID>0</rasd:InstanceID>
    <rasd:ResourceType>4</rasd:ResourceType>
    <rasd:VirtualQuantity>512</rasd:VirtualQuantity>
  </Item>

  <Item ovf:configuration="big">
    <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
    <rasd:ElementName>512 MB memory size &256 MB
reservation</rasd:ElementName>
    <rasd:InstanceID>0</rasd:InstanceID>
    <rasd:Reservation>256</rasd:Reservation>
  </Item>
</VirtualHardwareSection>
```

The resulting CIM instance for the resource allocation is illustrated below.

```
<rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
<rasd:ElementName>512 MB memory size & 256 MB reservation </rasd:ElementName>
<rasd:InstanceID>0</rasd:InstanceID>
<rasd:Reservation>256</rasd:Reservation>
<rasd:ResourceType>4</rasd:ResourceType>
<rasd:VirtualQuantity>512</rasd:VirtualQuantity>
```

The text highlighted in gray is from the "big" configuration. The text highlighted in blue is from the default configuration.

The following example illustrates a `DeploymentOptionsSection` element with three `Configuration` elements. Note that the "normal" configuration is designated as the default, so if no configuration is selected, the deployment uses the elements with a `configuration` attribute of "normal".

```
<DeploymentOptionSection>
    <Configuration ovf:id="minimal">
        <Label>Minimal</Label>
        <Description>Smallest practical implementation</Description>
    </Configuration>
    <Configuration ovf:id="normal" ovf:default="true">
        <Label>Normal</Label>
        <Description>A typical implementation</Description>
    </Configuration>
    <Configuration ovf:id="large">
        <Label>Large</Label>
        <Description>A scaled up implementation</Description>
    </Configuration>
</DeploymentOptionSection>
```

The following example illustrates the use of the `configuration` attribute to define the resource allocations for the three configurations above. The `VirtualHardwareSection` element uses the `CIM_ResourceAllocationDespcriptor` properties to specify the desired memory resource configuration. Each `Item` element in the `VirtualHardwareSection` refers to a configuration defined in the `DeploymentOptionsSection` element above.

```
<VirtualHardwareSection>
  <Info>...</Info>
  <Item ovf:configuration=”normal”>
      <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
      <rasd:ElementName>512 MB memory size and 256 MB reservation</rasd:ElementName>
      <rasd:InstanceID>0</rasd:InstanceID>
      <rasd:Reservation>256</rasd:Reservation>
      <rasd:ResourceType>4</rasd:ResourceType>
      <rasd:VirtualQuantity>512</rasd:VirtualQuantity>
   </Item>
   <Item ovf:configuration=”minimal”>
      <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
      <rasd:ElementName>256 MB memory size and 128 MB reservation</rasd:ElementName>
      <rasd:InstanceID>1</rasd:InstanceID>
      <rasd:Reservation>128</rasd:Reservation>
      <rasd:ResourceType>4</rasd:ResourceType>
      <rasd:VirtualQuantity>256</rasd:VirtualQuantity>
   </Item>
   <Item ovf:configuration="large">
      <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
      <rasd:ElementName>1024 MB memory size and 512 MB reservation</rasd:ElementName>
      <rasd:InstanceID>0</rasd:InstanceID>
      <rasd:Reservation>512</rasd:Reservation>
      <rasd:ResourceType>4</rasd:ResourceType>
      <rasd:VirtualQuantity>1024</rasd:VirtualQuantity>
   </Item>
</VirtualHardwareSection>
```

In the example above, the memory size is controlled by the configuration selected during deployment. If a configuration is not selected, the memory size defaults to the “normal” configuration.

The following example shows the use of a `configuration` attribute in a `ProductSection`.

```
<ProductSection>
   <Property ovf:key="app_log" ovf:type="string" ovf:value="low"
             ovf:configuration=”normal”>
        <Label>Loglevel</Label>
        <Description>Loglevel for the service</Description>
        <Value ovf:value="none" ovf:configuration="minimal">
        <Value ovf:value="high" ovf:configuration="large">
   </Property>
</ProductSection>
```

In this example, the value of “`app_log`” changes based on the configuration selected. As with virtual hardware, if a configuration is not selected, the value of the “`app_log`” property defaults to the “`normal`” configuration value, i.e., “`low`”.

4.4.7 SharedDiskSection element

The `SharedDiskSection` element allows a virtual disk to be referenced by multiple virtual systems to satisfy the needs of clustered databases. The file sharing system technology used is platform specific.

The `SharedDiskSection` element is a valid only at the envelope level.

Each shared disk has a unique identifier for the OVF package. The `SharedDiskSection` element adds a Boolean `ovf:readOnly` attribute that indicates whether read-write (i.e., `FALSE)`, or read-only (i.e., `TRUE`) access is allowed.

The following example illustrates the basics of a `SharedDiskSection` element.

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

The following example illustrates the use of shared disks. The disks for installations of the operating system (system), the cluster software (crs_home), and database (db_home) are backed by external `File` references. The shared virtual disks in this example have no backing by an external `File` reference; the deployment engine creates the shared disk appropriately to be shared by more than one virtual system.

```
<ovf:References>
  <ovf:File ovf:id="system" ovf:href="system.img" ovf:compression="gzip"/>
  <ovf:File ovf:id="crs_home" ovf:href="crs_home.img" ovf:compression="gzip"/>
  <ovf:File ovf:id="db_home" ovf:href="db_home.img" ovf:compression="gzip"/>
</ovf:References>
<ovf:DiskSection>
  <ovf:Info>Virtual Disks</ovf:Info>
  <ovf:Disk ovf:diskId="system" ovf:fileRef="system" ovf:capacity="5368709120"
ovf:format="Raw disk image"/>
  <ovf:Disk ovf:diskId="crs_home" ovf:fileRef="crs_home"  ovf:capacity="2147483648"
ovf:format="Raw disk image"/>
  <ovf:Disk ovf:diskId="db_home" ovf:fileRef="db_home"  ovf:capacity="4294967296"
ovf:format="Raw disk image"/>
```

```
</ovf:DiskSection>
<ovf:SharedDiskSection>
  <ovf:Info>Virtual Disks shared at runtime</ovf:Info>
  <ovf:SharedDisk ovf:diskId="crs_asm" ovf:capacity="4294967296" ovf:format="Raw disk 
image"/>
  <ovf:SharedDisk ovf:diskId="db_asm" ovf:capacity="12884901888" ovf:format="Raw disk 
image"/>
</ovf:SharedDiskSection>
.....
<ovf:VirtualSystemCollection ovf:id="rac_db_asm">
  <ovf:Info>Sample Oracle RAC using ASM</ovf:Info>
  .....
  <ovf:ScaleOutSection ovf:id="rac_db">
    <ovf:Info>RAC DB</ovf:Info>
    <ovf:Description>Number of instances</ovf:Description>
    <ovf:InstanceCount ovf:default="2" ovf:minimum="2" 
ovf:maximum="4"</ovf:InstanceCount>
  </ovf:ScaleOutSection>
  .....
  <ovf:VirtualSystem ovf:id="rac_db">
    <ovf:Info>RAC DB Instance</ovf:Info>
    .....
    <ovf:VirtualHardwareSection>
      <ovf:Info>System requirements: 8192 MB, 2 CPUs, 5 disks, 2 nics
      </ovf:Info>
      .....
      <ovf:Item>
        <rasd:Description>Disk 1</rasd:Description>
        <rasd:ElementName>Disk 1</rasd:ElementName>
        <rasd:HostResource>ovf:/disk/system</rasd:HostResource>
        <rasd:ResourceType>17</rasd:ResourceType>
      </ovf:Item>
      <ovf:Item>
        <rasd:Description>Disk 2</rasd:Description>
        <rasd:ElementName>Disk 2</rasd:ElementName>
        <rasd:HostResource>ovf:/disk/crs_home</rasd:HostResource>
        <rasd:ResourceType>17</rasd:ResourceType>
      </ovf:Item>
      <ovf:Item>
        <rasd:Description>Disk 3</rasd:Description>
        <rasd:ElementName>Disk 3</rasd:ElementName>
        <rasd:HostResource>ovf:/disk/db_home</rasd:HostResource>
        <rasd:ResourceType>17</rasd:ResourceType>
      </ovf:Item>
      <ovf:Item>
        <rasd:Description>Disk 4</rasd:Description>
        <rasd:ElementName>Disk 4</rasd:ElementName>
        <rasd:HostResource>ovf:/disk/crs_asm</rasd:HostResource>
        <rasd:ResourceType>17</rasd:ResourceType>
      </ovf:Item>
      <ovf:Item>
        <rasd:Description>Disk 5</rasd:Description>
        <rasd:ElementName>Disk 5</rasd:ElementName>
        <rasd:HostResource>ovf:/disk/db_asm</rasd:HostResource>
        <rasd:ResourceType>17</rasd:ResourceType>
      </ovf:Item>
      .....
    </ovf:VirtualHardwareSection>
  </ovf:VirtualSystem>
</ovf:VirtualSystemCollection>
```

4.4.8 PlacementGroupSection element

The `PlacementGroupSection` element defines an `ID` for a placement group and its associated placement policy(s). The placement group is associated with a `VirtualSystemCollection` or `VirtualSystem` through the use of the `PlacementSection` element.

An example of `PlacementGroupSection` elements is shown below.

```
  <ovf:PlacementGroupSection ovf:id="PG2" ovf:policy="availability">
    <Info>Placement policy for group of virtual systems that need availability</Info>
    <ovf:Description>Placement policy for a database tier</ovf:Description>
  </ovf:PlacementGroupSection>
      ...
  <ovf:PlacementGroupSection ovf:id="PG1" ovf:policy="affinity">
    <Info>Placement policy for group of virtual systems that need affinity</Info>
    <ovf:Description>Placement policy for a web tier</ovf:Description>
  </ovf:PlacementGroupSection>
```

The `PlacementGroupSection` element is a direct child element of the `Envelope` element. See 4.5.5.

4.5 OVF section elements used in Virtual System and Virtual System Collection

The following OVF descriptor section elements may appear within a `VirtualSystem` or `VirtualSystemCollection` element.

4.5.1 AnnotationSection element

The `AnnotationSection` element is user-defined and can appear in `VirtualSystem` and `VirtualSystemCollection` elements. An `AnnotationSection` element contains one `Annotation` element. `Annotation` elements are localizable. A suggested use for `Annotation` elements is to display them to the consumers as the package is deployed. Note that the `Annotation` element specified in OVF is not an XML Schema `Annotation` element.

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

4.5.2 ProductSection element

The `ProductSection` element 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 are be configured at installation time of the appliance, typically by prompting the user. This is discussed in more detail below.

```
<ProductSection ovf:class="com.mycrm.myservice" ovf:instance="1">
   <Info>Describes product information for the service</Info>
   <Product>MyCRM Enterprise</Product>
   <Vendor>MyCRM Corporation</Vendor>
   <Version>4.5</Version>
   <FullVersion>4.5-b4523</FullVersion>
   <ProductUrl>http://www.mycrm.com/enterprise</ProductUrl>
   <VendorUrl>http://www.mycrm.com</VendorUrl>
   <Icon ovf:height="32" ovf:width="32" ovf:mimeType="image/png" ovf:fileRef="icon">
   <Category>Email properties</Category>
   <Property ovf:key="adminEmail" ovf:type="string" ovf:userConfigurable="true">
        <Label>Admin email</Label>
        <Description>Email address of administrator</Description>
   </Property>
   <Category>Admin properties</Category>
   <Property ovf:key="appLog" ovf:type="string" ovf:value="low"
ovf:userConfigurable="true">
        <Description>Loglevel for the service</Description>
   </Property>
   <Property ovf:key="appisSecondary" ovf:value="false" ovf:type="boolean">
        <Description>Cluster setup for application server</Description>
   </Property>
   <Property ovf:key="appIp" ovf:type="string" ovf:value="${appserver-vm}">
        <Description>IP address of the application server VM</Description>
   </Property>
</ProductSection>
```

Note that the `ovf:key` attribute does not contain the period character ('.') or the colon character (':') and the `ovf:class` and `ovf:instance` attributes do not contain the colon character (':').

If only one instance of a product is installed, the `ovf:instance` attribute is not used.

The following illustrates the use of OVF `Properties` in a `ProductSection` element:

```
<ProductSection>
   <Property ovf:key="adminEmail" ovf:type="string" ovf:userConfigurable="true"
             ovf:configuration="standard">
        <Label>Admin email</Label>
        <Description>Email address of service administrator</Description>
   </Property>
   <Property ovf:key="appLog" ovf:type="string" ovf:value="low"
             ovf:userConfigurable="true">
        <Label>Loglevel</Label>
        <Description>Loglevel for the service</Description>
        <Value ovf:value="none" ovf:configuration="minimal">
   </Property>
</ProductSection>
```

In the example above, the `adminEmail` property is only user configurable in the standard configuration, while the default value for the `appLog` property is changed from "low" to "none" in the minimal configuration.

4.5.3 EulaSection element

The `EulaSection` contains the human readable licensing agreement for its parent, usually a `VirtualSystem` or `VirtualSystemCollection` element. Each parent element may have more than one `EulaSection element.` The contents of the `License` element of each `EulaSection` element are displayed to the user for acceptance when the OVF package is deployed. If unattended deployment is supported, provision is made for implicit acceptance of the Eula.

Eulas may be externalized for localization or to point to an external license document. See 5.2 for more details about internationalization.

This is an example `EulaSection` element:

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

4.5.4 VirtualHardwareSection element

The `VirtualHardwareSection` element describes the virtual hardware that is using the CIM resource allocation setting data model. This model is based on `CIM_ResourceAllocationSettingData` classes that specify CIM properties to describe the type and quantity of the resource being requested. The CIM Schema is available at http://www.dmtf.org/standards/cim.

The minimum required resource allocation setting data for a virtual processor device is illustrated below. This is not user friendly, so it helps to add the `rasd:Description` and `rasd:ElementName`.

```
<Item>
  <rasd:InstanceID>0</epasd:InstanceID>
  <rasd:ResourceType>3</epasd:ResourceType>
</Item>
```

This `VirtualHardwareSection` element describes the virtual devices in the hardware abstraction layer that is used by a virtual system. The `CIM_ResourceAllocationSettingData` has a list of devices. Some devices, such as the Ethernet port and Storage, are subclassed with an extended set of properties.

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. An incomplete or missing hardware section may cause the deployment to fail.

The following illustrates a `VirtualHardwareSection` element.

```
<VirtualHardwareSection>
  <Info>Memory = 4 GB, CPU = 1 GHz, Disk = 100 GB, 1 Ethernet nic</Info>
  <Item>
    <rasd:AllocationUnits>Hertz*10^9</rasd:AllocationUnits>
    <rasd:Description>Virtual CPU</rasd:Description>
    <rasd:ElementName>1 GHz virtual CPU</rasd:ElementName>
    <rasd:InstanceID>1</rasd:InstanceID>
    <rasd:Reservation>1</rasd:Reservation>
    <rasd:ResourceType>3</rasd:ResourceType>
    <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
    <rasd:VirtualQuantityUnit>Count</ rasd:VirtualQuantityUnit>
  </Item>
  <Item>
    <rasd:AllocationUnits>byte*2^30</rasd:AllocationUnits>
    <rasd:Description>Memory</rasd:Description>
    <rasd:ElementName>1 GByte of memory</rasd:ElementName>
    <rasd:InstanceID>2</rasd:InstanceID>
    <rasd:Limit>4</rasd:Limit>
    <rasd:Reservation>41</rasd:Reservation>
    <rasd:ResourceType>4</rasd:ResourceType>
  </Item>
  <EthernetPortItem>
    <epasd:AllocationUnits>bit / second *2^30 </rasd:AllocationUnits>
    <epasd:Connection>VM Network</epasd:Connection>
    <epasd:Description>Virtual NIC</epasd:Description>
    <epasd:ElementName>Ethernet Port</epasd:ElementName>
    <epasd:NetworkPortProfileID>1</epasd:NetworkPortProfileID>
    <epasd:NetworkPortProfileIDType>4</epasd:NetworkPortProfileIDType>
    <epasd:ResourceType>10</epasd:ResourceType>
    <epasd:VirtualQuantity>1</epasd:VirtualQuantity>
    <epasd:VirtualQuantityUnits>Count</epasd:VirtualQuantityUnits>
  </EthernetPortItem>
  <StorageItem>
    <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
    <sasd:Description>Virtual Disk</sasd:Description>
    <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>

    <sasd:Reservation>100</sasd:Reservation>
    <sasd:ResourceType>31</sasd:ResourceType>
    <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
    <sasd:VirtualQuantityUnit>Count</sasd:VirtualQuantityUnit>
  </StorageItem>
</VirtualHardwareSection>
```

An example of a `ResourceSubType` CIM property follows:

```
    <rasd:ResourceSubType>buslogic lsilogic</rasd:ResourceSubType>
```

The following example illustrates a `VirtualHardwareSection` element with a default and a ‘`big`’ configuration. See 4.4.6 for information about how configuration options are used.

```
<VirtualHardwareSection>
  <Info>...</Info>
  <Item>
    <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
    <rasd:ElementName>512 MB memory size and 256 MB reservation</rasd:ElementName>
    <rasd:InstanceID>0</rasd:InstanceID>
    <rasd:Reservation>256</rasd:Reservation>
    <rasd:ResourceType>4</rasd:ResourceType>
    <rasd:VirtualQuantity>512</rasd:VirtualQuantity>
  </Item>
   ...
  <Item ovf:configuration="big">
    <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
    <rasd:ElementName>1024 MB memory size and 512 MB reservation</rasd:ElementName>
    <rasd:InstanceID>0</rasd:InstanceID>
    <rasd:Reservation>512</rasd:Reservation>
    <rasd:ResourceType>4</rasd:ResourceType>
    <rasd:VirtualQuantity>1024</rasd:VirtualQuantity>
  </Item>
</VirtualHardwareSection>
```

4.5.5 PlacementSection element

The `PlacementSection` element specifies the placement group of which a Virtual System or Virtual System Collection is a member. The placement policy(s) specified in the placement group (see 4.4.8) is applied by the deployment function. The following OVF descriptor snippet illustrates a placement section in each of two virtual systems.

```
  <VirtualSystemCollection ovf:id="VSC10">
    <VirtualSystem ovf:id="VS11">
      <Info>Web server</Info>
      ...
      <ovf:PlacementSection ovf:group="PG1">
        <Info>Placement policy group reference</Info>
      </ovf:PlacementSection>
      ...
    </VirtualSystem>
    <VirtualSystem ovf:id="VS21">
      <Info>Web server</Info>
      ...
      <ovf:PlacementSection ovf:group="PG1">
        <Info>Placement policy group reference</Info>
      </ovf:PlacementSection>
      ...
    </VirtualSystem>
  </VirtualSystemCollection>
```

In this example, the virtual systems, when instantiated, should be placed according to the placement policies specified in the "`PG1`" placement group. As shown in 4.4.8, the placement group '`PG1`' has an affinity placement policy. Figure 7 illustrates an affinity placement.

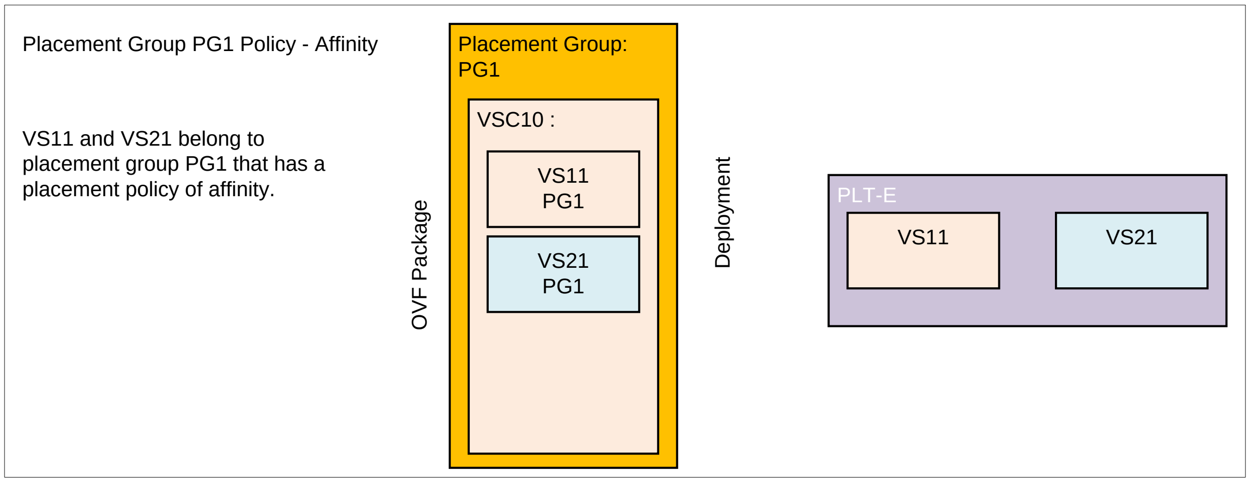

Placement Group PG1 Policy - Affinity
VS11 and VS21 belong to placement group PG1 that has a placement policy of affinity.
OVF Package
Placement Group: PG1
VSC10 :
VS11 PG1
VS21 PG1
Deployment
PLT-E
VS11
VS21

Figure 7 – Affinity placement

The following OVF descriptor snippet illustrates a placement section in each of two virtual systems.

```
<VirtualSystemCollection ovf:id="VSC10">
  <VirtualSystem ovf:id="VS11">
    <Info>Web server</Info>
    ...
    <ovf:PlacementSection ovf:group="PG2">
      <Info>Placement policy group reference</Info>
    </ovf:PlacementSection>
    ...
  </VirtualSystem>
  <VirtualSystem ovf:id="VS21">
    <Info>Web server</Info>
    ...
    <ovf:PlacementSection ovf:group="PG2">
      <Info>Placement policy group reference</Info>
    </ovf:PlacementSection>
    ...
  </VirtualSystem>
</VirtualSystemCollection>
```

In this example, the virtual systems, when instantiated, should be placed according to the placement policies specified in the “`PG2`” placement group. As shown in 4.4.8, the placement group ‘`PG2`’ has an availability placement policy. Figure 8 illustrates an availability placement.

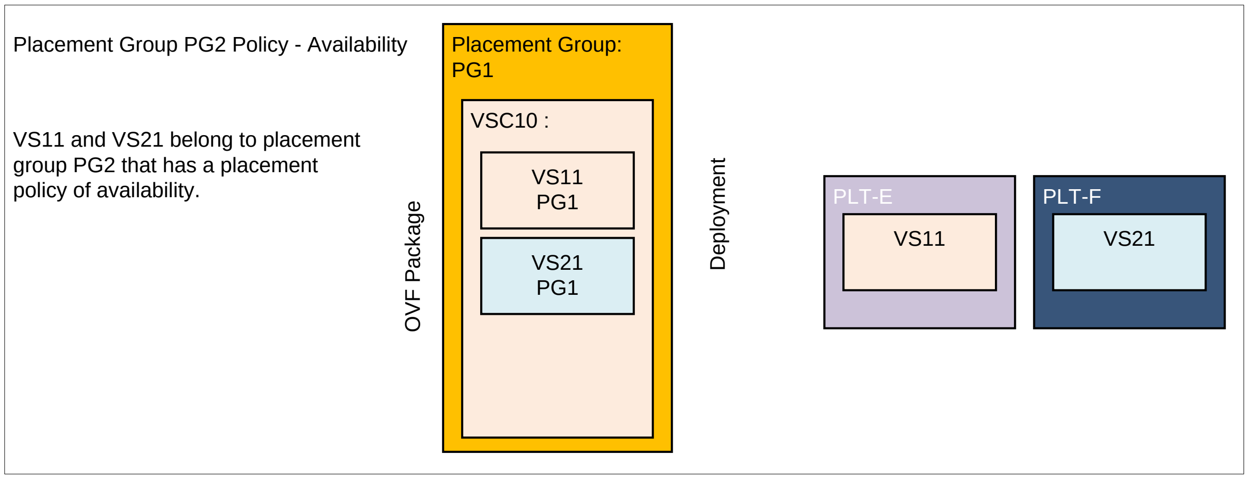

Placement Group PG2 Policy - Availability
VS11 and VS21 belong to placement group PG2 that has a placement policy of availability.
OVF Package
Placement Group: PG1
VSC10 :
VS11 PG1
VS21 PG1
Deployment
PLT-E
VS11
PLT-F
VS21

Figure 8 – Availability placement

The following OVF descriptor snippet illustrates a placement section in each of two virtual systems.

```
  <VirtualSystemCollection ovf:id="VSC10">
      <ovf:PlacementSection ovf:group="PG1">
        <Info>Placement policy group reference</Info>
      </ovf:PlacementSection>

    <VirtualSystem ovf:id="VS11">
      <Info>Web server</Info>
      ...
    </VirtualSystem>
    <VirtualSystem ovf:id="VS21">
      <Info>Web server</Info>
      ...
      <ovf:PlacementSection ovf:group="PG2">
        <Info>Placement policy group reference</Info>
      </ovf:PlacementSection>
      ...
    </VirtualSystem>
   <VirtualSystem ovf:id="VS31">
      <Info>Web server</Info>
      ...
    </VirtualSystem>
    <VirtualSystem ovf:id="VS41">
      <Info>Web server</Info>
      ...
      <ovf:PlacementSection ovf:group="PG2">
        <Info>Placement policy group reference</Info>
      </ovf:PlacementSection>
      ...
    </VirtualSystem>
  </VirtualSystemCollection>
```

In this example, the virtual systems, when instantiated, should be placed according to the placement policies specified in the "`PG1`" and "`PG2`" placement groups. As shown in 4.4.8, the placement group '`PG1`' has an affinity placement policy. As shown in 4.4.8, the placement group '`PG2`' has an availability placement policy. Figure 9 illustrates the placement of virtual systems in this more complex example. It also illustrates the application of the transitivity rule.

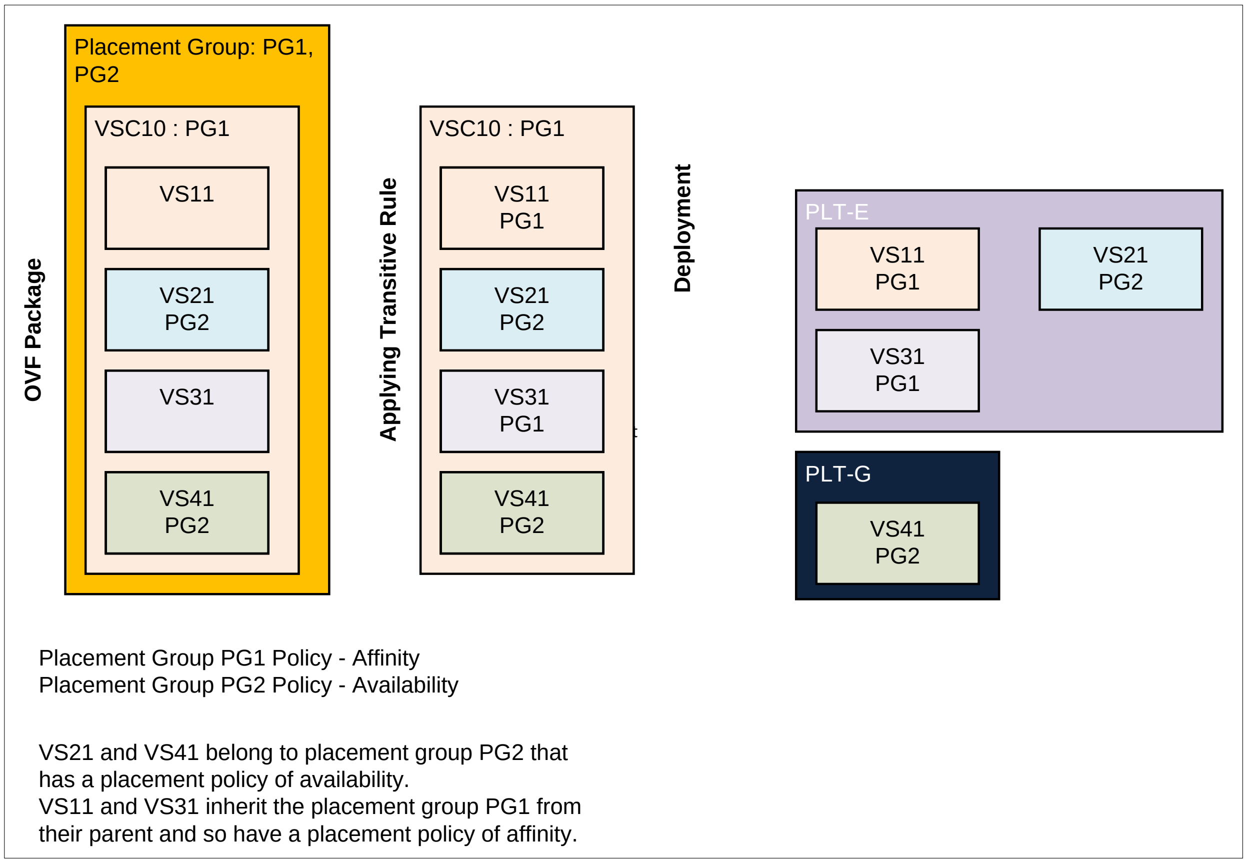

Placement Group: PG1, PG2
VSC10 : PG1
VS11
VS21
PG2
VS31
VS41
PG2
OVF Package
Applying Transitive Rule
VSC10 : PG1
VS11
PG1
VS21
PG2
VS31
PG1
VS41
PG2
Deployment
PLT-E
VS11
PG1
VS21
PG2
VS31
PG1
PLT-G
VS41
PG2
Placement Group PG1 Policy - Affinity
Placement Group PG2 Policy - Availability
VS21 and VS41 belong to placement group PG2 that has a placement policy of availability.
VS11 and VS31 inherit the placement group PG1 from their parent and so have a placement policy of affinity.

Figure 9 – Affinity and availability placement

4.5.6 EncryptionSection element

There are several reasons why it is desirable to have an encryption scheme enabling exchange of OVF appliances while ensuring that only the intended consumers can use them. The encryption scheme proposed in this specification utilizes existing encryption standards to incorporate this functionality in the specification,

The `EncryptionSection` element provides a single location for placing the encryption algorithm-related markup and the corresponding reference list to point to the OVF content that has been encrypted.

A document typically uses a single method of encryption, with a single key. However, the specification allows the flexibility to encrypt different portions of the OVF descriptor with different keys derived by using different methods and communicated to the end user in different ways.

It is important to keep in mind that depending on which parts of the OVF descriptor have been encrypted, an OVF descriptor may not validate against the OVF Schemas until decrypted.

The encryption uses XML Encryption standard 1.1 to encrypt either the files in the reference section or any parts of the XML markup of an OVF document.

From an encryption standpoint, the important aspects that the standard defines are the

a) algorithm used for the derivation of the key used in the encryption

b) block encryption algorithm used to encrypt the content that uses the key

c) method of transporting keys embedded in the OVF XML document.

For each method of encryption used within the document, all the aspects that are necessary need to be defined by the OVF package author. For instance, the author may choose to embed the key used in the document, or the author may choose to communicate the key to desired end user by other means.

The other aspect is a list of references to the markup sections in the OVF envelope, or the files in the reference section that are encrypted by using the specific method. In order to be able to encrypt arbitrary sections within the OVF Descriptor, use is made of the XML `ID` attribute within the `ReferenceList` elements.

The following example illustrates the conceptual structure of an `Encryption` section.

```
<! --- Start of encryption section ---!>
  <! ---- Start of Markup for encryption method 1 ----!>
    <! ---- Markup defining key derivation aspects per XML encryption 1.1 ----!>
    <! ---- Markup defining the usage of the key for encryption per XML encryption 1.1
---!>
    <! ---- Optionally, the markup for key transportation per XML encryption 1.1 ---!>
    <! ---- Start of markup for pointers to the list of XML fragments encrypted using
method 1---!>
        <! --- Pointer 1 ---!>
            .
            .
        <! --- Pointer N ---!>
    <! ---- End of markup for pointers to the list of XML fragments encrypted using
method 1 ---!>
  <! ---- End of Markup for method 1 of encryption ----!>

  <! ---- Start of the markup for encryption method N ----!>
        <! ---- Markup defining key derivation aspects per XML encryption 1.1 ----!>
        <! ---- Markup defining the usage of the key for encryption per XML encryption
1.1 ---!>
        <! ---- Optionally, the markup for key transportation per XML encryption 1.1 -
--!>
        <! ---- Start of markup for pointers to the list of XML fragments encrypted
using method 1---!>
        <! --- Pointer 1 ---!>
            .
            .
        <! --- Pointer N ---!>
  <! ---- End of Markup for encryption method N ----!>
<! --- End of encryption section ---!>
```

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

```
  <ovf:EncryptionSection>
<!--- This section contains two different methods of encryption and the corresponding
backpointers to the data that is encrypted. ->
  <!--- Method#1: Pass phrase based key derivation ->
```

```
<!--- The following derived key block defines PBKDF2 and the corresponding
backpointers to the encrypted data elements. -->
  <!--- Use a salt value “ovfpassword” and iteration count of 4096. --->
 <xenc11:DerivedKey>
       <xenc11:KeyDerivationMethod
Algorithm=”http://www.rsasecurity.com/rsalabs/pkcs/schemas/pkcs-5#pbkdf2”/>
<pkcs-5:PBKDF2-params>
          <Salt>
                 <Specified>ovfpassword</Specified>
             </Salt>
             <IterationCount>4096</IterationCount>
             <KeyLength>16</KeyLength>
             <PRF Algorithm=”http://www.w3.org/2001/04/xmldsig-more#hmac-sha256”/>
       </pkcs-5:PBKDF2-params>
…
<!—- The ReferenceList element below contains references to the file Ref-109.vhd via
the URI syntax that is specified by XML Encryption.
--->
<xenc:ReferenceList>
   <xenc:DataReference URI=”#first.vhd” />
<xenc:DataReference URI=… />
<xenc:DataReference URI=… />
</xenc:ReferenceList>
   </xenc11:DerivedKey>
    <!-- Method#2: The following example illustrates use of a symmetric key
transported by using the public key within a certificate. ->
<xenc:EncryptedKey>
       <xenc:EncryptionMethod    Algorithm=”http://www.w3.org/2001/04/xmlenc#rsa-
1_5”/>
          <ds:KeyInfo xmlns:ds=’http://www.w3.org/2000/09/xmldsig#’
             <ds:X509Data>
             <ds:X509Certificate> … </ds:X509Certificate>
       </ds:X509Data>
       </ds:KeyInfo>
       <xenc:CipherData>
   <xenc:CipherValue> … </xenc:CipherValue>
       </xenc:CipherData>
<!—- The ReferenceList element below contains reference #second-xml-fragment” to the
XML fragment that has been encrypted by using the above method. --->
   <xenc:ReferenceList>
       <xenc:DataReference URI=’#second-xml-fragment’ />
       <xenc:DataReference URI=’…’ />
       <xenc:DataReference URI=’…’ />
   </xenc:ReferenceList>
    </xenc:EncryptedKey>
  </ovf:EncryptionSection>
Below is an example of the encrypted file that is referenced in the EncryptionSection
above by using URI='Ref-109.vhd' syntax.
EXAMPLE:
<ovf:References>
<ovf:File ovf:id="Xen:9cb10691-4012-4aeb-970c-3d47a906bfff/0b13bdba-3761-8622-22fc-
```

```
2e252ed9ce14" ovf:href="Ref-109.vhd">
<!-- The encrypted file referenced by the package is enclosed by an EncryptedData with
a CipherReference to the actual encrypted file. The EncryptionSection in this example
has a backpointer to it under the PBKDF2 algorithm via Id=”first.vhd”. This tells the
decrypter how to decrypt the file. -->
<xenc:EncryptedData Id="first.vhd" Type='http://www.w3.org/2001/04/xmlenc#Element' >
                 <xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc" />
                    <xenc:CipherData>
                           <xenc:CipherReference URI='Ref-109.vhd'/>
                    </xenc:CipherData>
</xenc:EncryptedData>
</ovf:File>
</ovf:References>
Below is an example of the encrypted OVF markup that is referenced in the
EncryptionSection above by using URI=’#second-xml-fragment’ syntax.
EXAMPLE:
<!—-  The EncryptedData element below encompasses encrypted xml from the original
document. It is provided with the Id “second-xml-fragment”, which allows it to be
referenced from the EncryptionSection. -->
<xenc:EncryptedData Type=http://www.w3.org/2001/04/xmlenc#Element Id=”second-xml-
fragment”>
<!-- Each EncryptedData specifies its own encryption method. -->
   <xenc:EncryptionMethod Algorithm=http://www.w3.org/2001/04-xmlenc#aes128-cbc/>
   <xenc:CipherData>
      <!--- Encrypted content --->
      <xenc:CipherValue>DEADBEEF</xenc:CipherValue>
   </xenc:CipherData>
  </xenc:EncryptedData>
```

4.6 OVF section elements used in virtual system collection

4.6.1 ResourceAllocationSection element

The `ResourceAllocationSection` element sets resource constraints that apply to a virtual system collection. In contrast, the `VirtualHardwareSection` element applies to a specific virtual system.

The `ResourceAllocationSection` element can use the `bound` attribute to set a minimum, a maximum, or both for the allocation of a resource that applies in aggregate to all the virtual systems in the virtual system collection.

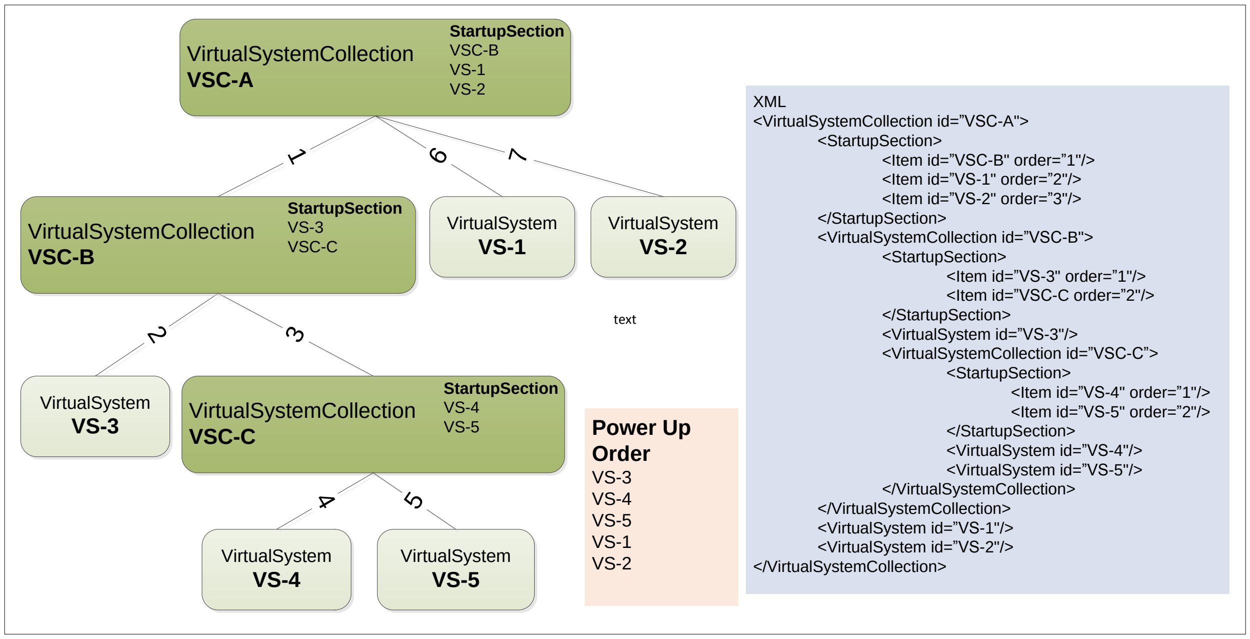
The following example illustrates a `ResourceAllocationSection` element. The processor allocation illustrates the use of the `bound` attribute.

```
<ResourceAllocationSection>
   <Info>Defines reservations for CPU and memory for the collection of VMs</Info>
   <Item>
      <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
      <rasd:ElementName>300 MB reservation</rasd:ElementName>
      <rasd:InstanceID>0</rasd:InstanceID>
      <rasd:Reservation>300</rasd:Reservation>
      <rasd:ResourceType>4</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>
```

4.6.2 StartupSection element

The `StartupSection` element controls powering on and off of virtual system collections and is executed after `InstallSection` element(s). The `StartupSection` element is a list of `Item` elements. `Item` elements have attributes that control the order and timing of power on and power off. The `Item` elements in a `StartupSection` element are scoped to that element. Do not confuse these elements with `Item` elements in a `VirtualHardwareSection` element or `ResourceAllocationSection` element.

The `Item` elements within a `StartupSection` element reference either a `VirtualSystem` element or a `VirtualSystemCollection` element. A `StartupSection` element may control powering up and down of both virtual systems and virtual system collections contained in the virtual system collection parent. This format allows for recursive startup structures. See Figure 10.



```
XML
<VirtualSystemCollection id=”VSC-A">
        <StartupSection>
                <Item id=”VSC-B" order=”1"/>
                <Item id=”VS-1" order=”2"/>
                <Item id=”VS-2" order=”3"/>
        </StartupSection>
        <VirtualSystemCollection id=”VSC-B">
                <StartupSection>
                        <Item id=”VS-3" order=”1"/>
                        <Item id=”VSC-C order=”2"/>
                </StartupSection>
                <VirtualSystem id=”VS-3"/>
                <VirtualSystemCollection id=”VSC-C”>
                        <StartupSection>
                                <Item id=”VS-4" order=”1"/>
                                <Item id=”VS-5" order=”2"/>
                        </StartupSection>
                        <VirtualSystem id=”VS-4"/>
                        <VirtualSystem id=”VS-5"/>
                </VirtualSystemCollection>
        </VirtualSystemCollection>
        <VirtualSystem id=”VS-1"/>
        <VirtualSystem id=”VS-2"/>
</VirtualSystemCollection>
```

Figure 10 – StartupSection traversal

The order of startup is determined by the value of the `order` attributes of the `Item` elements in a `StartupSection` element in a `VirtualSystemCollection` element. The `order` attribute is a nonnegative integer.

If the `order` attribute of an `Item` element has a value of `'0' (zero)`, the virtual system or virtual system collection may be powered up at any time. The virtualization platform does not have to wait to start items that have a higher order value.

If the `order` attribute of an `Item` element has a nonzero value, the virtual systems are started in ascending numeric order. Virtual systems that have the same value of the `order` attribute may be started concurrently.

It is recommended that virtual systems are stopped in descending numeric order. Virtual systems with `Item` elements with an `order` attribute value of '`0' (zero)` may be stopped at any time.

However, virtual systems are permitted to stop in a nondescending order in an implementation specific manner unless the `shutdownorder` attribute is specified. The `shutdownorder` attribute allows the shutdown order to be specified.

Several optional attributes of the `Item` element support more detailed control of starting and stopping. The `startDelay` and `stopDelay` attributes specify the seconds to wait until executing the next step in the sequence. The default for both `startDelay` and `stopDelay` is zero.

The `startAction` and `stopAction` attributes specify the actions to use in starting and stopping. Valid values for `startAction` are `powerOn` and `none.` The default value is `'powerOn'.` Valid values for the `stopAction` attribute are `'powerOn'`, `'guestShutdown'`, and `'none'.` The default value is `'powerOff'`. If the `stopAction` attribute is set to `'guestShutdown'`, the action taken is deployment platform specific.

The `waitingForGuest` attribute is a Boolean that allows the deployment platform to wait until the guest software reports readiness. The default value is `'FALSE'`. The communication mechanism is platform specific.

The following example illustrates a `StartupSection` element.

```
<StartupSection>
  <Item ovf:id="vm1" ovf:order="0" ovf:startDelay="30" ovf:stopDelay="0"
    ovf:startAction="powerOn" ovf:waitingForGuest="true" ovf:stopAction="powerOff"/>
  <Item ovf:id="teamA" ovf:order="0"/>
  <Item ovf:id="vm2" ovf:order="1" ovf:startDelay="0" ovf:stopDelay="20"
     ovf:startAction="powerOn" ovf:stopAction="guestShutdown"/>
</StartupSection>
```

4.6.3 ScaleOutSection element

The `ScaleOutSection` element allows dynamic configuration of the number of instantiated virtual systems in a `VirtualSystemCollection` element. Without a `ScaleOut` element in the `VirtualSystemCollection` element, the number of virtual systems and virtual system collections is fixed. The `ScaleOutSection` element specifies a minimum and maximum number of replicas to be created. At deployment time, the deployment platform chooses a value between the minimum and maximum `InstanceCount.` The consumer can be queried for the value or the deployment platform can make a determination based on other metadata. The `ScaleOutSection` element only appears in `VirtualSystemCollection` elements, although both virtual system and virtual system collections may be replicated.

The following example illustrates a `ScaleOutSection` element.

```
<VirtualSystemCollection ovf:id="web-tier">
  ...
  <ovf:ScaleOutSection ovf:id="web-server">
    <Info>Web tier</Info>
    <ovf:Description>Number of web server instances in web tier</ovf:Description>
    <ovf:InstanceCount ovf:default="4" ovf:minimum="2" ovf:maximum="8"/>
  </ovf:ScaleOutSection>
  ...
  <VirtualSystem ovf:id="web-server">
    <Info>Prototype web server</Info>
    ...
  </VirtualSystem>
</VirtualSystemCollection>
```

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

```
<Environment ... ovfenv:id=”web-server-1”>
  ...
  <Entity ovfenv:id=”web-server-2”>
    ...
  </Entity>
  <Entity ovfenv:id=”web-server-3”>
    ...
  </Entity>
</Environment>
```

Note that the OVF `id` of the replicas is derived from the `id` of the prototype virtual system by adding a sequence number. After deployment, all replica virtual systems have a sequence number suffix and no virtual system has the base `id` of the prototype. If there is a `StartupSection` element, each replica has the same startup number. It is not possible to specify a startup order among replicas.

EXAMPLE:

```
<VirtualSystemCollection ovf:id="web-tier">
  ...
  <DeploymentOptionSection>
    <Info>Deployment size options</Info>
    <Configuration ovf:id="minimal">
      <Label>Minimal</Label>
      <Description>Minimal deployment scenario</Description>
    </Configuration>
    <Configuration ovf:id="common" ovf:default="true">
      <Label>Typical</Label>
      <Description>Common deployment scenario</Description>
    </Configuration>
    ...
  </DeploymentOptionSection>
  ...
  <ovf:ScaleOutSection ovf:id="web-server">
    <Info>Web tier</Info>
    <ovf:Description>Number of web server instances in web tier</ovf:Description>
      <ovf:InstanceCount ovf:default="4"/>
      <ovf:InstanceCount ovf:default="1" ovf:configuration="minimal"/>
  </ovf:ScaleOutSection>
...
</VirtualSystemCollection>
```

In the example above, a `DeploymentOptionSection` element is used to control values for the `InstanceCount` element in a `ScaleOutSection` element. Values in a `ScaleOutSection` element can also be controlled through OVF `property` elements. The OVF properties are prompted for one time for each replica. If the author wants an OVF property to be shared among replicas, it can be placed within the `VirtualSystemCollection` element.

4.7 OVF section elements used in virtual system

4.7.1 OperatingSystemSection element

The `OperatingSystemSection` element specifies the guest operating system used in a virtual system. The selection of operating systems comes from the `CIM_OperatingSystem.OSType` property. The OVF `version` and OVF `id` correspond to the `Value` and `ValueMap` of that property.

The `id` attribute is required and refers to an integer from the `ValueMap`. The `version` attribute is optional and refers to the corresponding `Value` for the `ValueMap`. The `version` attribute is a symbolic string and cannot be internationalized.

Both the `Info` (derived from Section) and `Description` elements may be externalized for localization. See 5.2.

This example is of a section that specifies a Microsoft Windows Server 2008:

```
<OperatingSystemSection ovf:id="76" ovf:version="Microsoft Windows Server 2008">
   <Info>Specifies the operating system installed</Info>
   <Description>Microsoft Windows Server 2008</Description>
</OperatingSystemSection>
```

4.7.2 InstallSection element

The `InstallSection` element is optional and is used only in `VirtualSystem` elements. If present, it is processed before the `StartupSection` element.

The `InstallSection` elements enable the OVF package author to specify that a virtual system needs to reboot before powering off in order to complete the installation. Typically, when the boot occurs, the guest software executes scripts or other software from the OVF environment to complete the installation. The absence of an `InstallSection` element implies that a boot is not necessary to complete the installation. For example, if the virtual system has no guest software or the guest software is installed in the system image, an `InstallSection` element is not needed.

The virtual systems in a virtual system collection may each have an `InstallSection` element defined. The reboots may be concurrent.

The value of the `initialBootStopDelay` attribute is the duration in seconds that the virtualization platform waits for the virtual system to power off. If the delay expires and the virtual system has not powered off, the installation is deemed to have failed. The default value for `initialBootStopDelay` is zero, meaning that there is no limit on the delay and the virtualization platform waits until the virtual system powers itself off. The guest software on the virtual system could boot multiple times before powering off.

In the example below, the virtualization platform waits 5 minutes (300 seconds) for the guest software to power off the virtual system. If the virtual machine does not power off in 5 minutes, the installation is deemed a failure. During the 5-minute wait interval, the virtual system could reboot several times.

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

4.7.3 EnvironmentFilesSection element

The `EnvironmentFilesSection` element allows the conveyance of additional environment files to the guest software permitting additional customization. These files are conveyed by using the same transport media as the OVF environment file.

The OVF environment file is generated by the deployment function; however, any additional environment files are not and must be provided by the OVF package author. The additional environment files are specified in the `EnvironmentFilesSection` element with a `File` element that has an `ovf:fileRef` attribute and `ovf:path` attribute for each file.

The `ovf:fileRef` attribute points to a File element in the `References` element. The `File` element is identified by matching its `ovf:id` attribute value with the `ovf:fileRef` attribute value.

The `ovf:path` attribute indicates the relative location in the transport media where the file is placed.

```
<Envelope>
  <References>
    ...
  <File ovf:id="config" ovf:href="config.xml" ovf:size="4332"/>
  <File ovf:id="resources" ovf:href="http://mywebsite/resources/resources.zip"/>
  </References>
  ...
  <VirtualSystem ovf:id="...">
```

```
    ...
  <ovf:EnvironmentFilesSection ovf:required="false" ovf:transport="iso">
    <Info>Config files to be included in OVF environment</Info>
    <ovf:File ovf:fileRef="config" ovf:path="setup/cfg.xml"/>
    <ovf:File ovf:fileRef="resources" ovf:path="setup/resources.zip"/>
  </ovf:EnvironmentFilesSection>
 ...
  </VirtualSystem>
  ...
</Envelope>
```

In the example above, the file `config.xml` in the OVF package is copied to the OVF environment ISO image and is accessible to the guest software in location `/ovffiles/setup/cfg.xml`, while the file `resources.zip` is accessible in location `/ovffiles/setup/resources.zip` at deployment.

4.7.4 BootDeviceSection element

Earlier versions of OVF allowed virtual systems to boot only from the default boot device. This was found to be a limitation in various scenarios that are encountered in OVF deployment.

a) There was no way to specify whether a virtual system needed to be set up to PXE boot from a NIC. Also, there was no way to specify whether a virtual system needed to be set up to boot from a secondary disk or a USB device. Thus, there was a need to be able to specify these alternative boot sources with their corresponding settings.

b) A further need was identified, through implementation experience, to be able to specify multiple alternative boot configurations. For instance during the “preparation” phase of the OVF, it may be necessary for a virtual system to be patched by using a fix-up disk.

The Common Information Model (CIM) defines artifacts to deal with boot order use cases prevalent in the industry for BIOSes found in desktops and servers. The `CIM_BootSourceSetting` class defines an individual boot source device, like an NIC or a disk, that is to be used as the boot source. Each of the devices is identified by a unique `ID` specified in the `CIM_BootSourceSetting` class.

A boot configuration is defined by a sequence of boot devices under an aggregation class, `CIM_BootConfigSetting`. Thus a sequence of one or more `CIM_BootSourceSetting` properties is aggregated into the `CIM_BootConfigSetting` class.

The OVF envelope allows multiple such boot configurations to be aggregated into the `BootDeviceSection`.element. Each such `BootDeviceSection` element can be part of a `VirtualHardwareSection` element.

A deployment function attempts to set up a boot source sequence for a virtual system as defined in the boot configuration that it has chosen. Choosing a boot configuration is an issue if there are more than one boot configurations. The deployment function makes that choice based on the state of the deployment and the caption element in the boot configuration structure.

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

```
EXAMPLE:
  <Envelope>
  ...
  <VirtualSystem ovf:id="...">
    ...
    <ovf:BootDeviceSection>
      <Info>Boot device order specification</Info>
      <bootc:CIM_BootConfigSetting>
        <bootc:Caption>Pre-Install</bootc:Caption>
        <bootc:Description>Boot Sequence for fixup of disk</bootc:Description>
        <boots:CIM_BootSourceSetting>
          <boots:Caption>Fix-up DVD on the network</boots:Caption>
          <boots:InstanceID>3</boots:InstanceID>           <!— Network device-->
        </boots:CIM_BootSourceSetting>
        <boots:CIM_BootSourceSetting>
          <boots:Caption>Boot virtual disk</boots:Caption>
          <boots:StructuredBootString>CIM:Hard-Disk</boots:StructuredBootString>
        </boots:CIM_BootSourceSetting>
      </bootc:CIM_BootConfigSetting>
    </ovf:BootDeviceSection>
    ...
  </VirtualSystem>
</Envelope>
```

5 Authoring an OVF package

5.1 Creation

The creation of an OVF package involves

i) packaging a set of VMs onto a set of virtual disks

ii) encoding of those virtual disks appropriately

iii) attaching an OVF descriptor with a specification of the virtual hardware, licensing, and other customization metadata

iv) possibly including the attachment of a digital signature for the package

The process of deploying an OVF package 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 metadata needed for correct installation and execution. This transforms the virtual machine from its current run-time state on a particular hypervisor into an OVF package. During this process, the virtual machine's disks can be compressed to make them more convenient to distribute.

For commercial-grade virtual appliances, a standard build environment can be used to produce an OVF package. For example, the OVF descriptor can be managed by using a source control system, and the OVF package can be built by 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 package is created, it can 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 ordered shutdown.

The process of authoring the OVF descriptor is essentially putting together the building blocks for the virtual appliance. As indicated earlier, a virtual appliance is defined by the description of the virtual systems composing the appliance, metadata regarding the appliance and the guest software, and a set of referenced files. The OVF descriptor is the central element to aggregate and reference all required information. The major building blocks of the OVF descriptor are sections. Clause 4 introduces the various sections that can be used to describe the virtual appliance.

5.2 Internationalization

The OVF specification supports localizable messages by using the optional `ovf:msgid` attribute. Localized messages can be used to display the user messages in the local language during deployment.

```
<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>
```

The example above defines an `Info` element within a section. The information in this section is related to the operating system of the virtual system. The attribute `ovf:msgid="info.os"` indicates that the string between the start-tag and the end-tag of the `Info` element can be replaced with a localized message. The localized message is referred to by its message ID of `info.os`. If there is a suitable localized message set in a `Strings` section, the default message “Operating System” is replaced by the localized message taken from the `Strings` section corresponding to the current region.

In the example above the localized strings are stored inside the OVF descriptor. Localized strings can also be stored outside the OVF descriptor by using external string bundles. For example:

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

The localized message for “Operating System” is defined in the files `danish.msg` and `german.msg`. The format of the external message file `german.msg` is described in the example below.

```
<Strings
 xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1"
 xmlns="http://schemas.dmtf.org/ovf/envelope/1"
 xml:lang="de-DE">
   ...
   <Msg ovf:msgid="info.os">Betriebssystem</Msg>
   ...
</Strings>
```

In the top `Strings` section, the `xml:lang` attribute is used to define the locale of the particular external message file. The external message file contains `Msg` elements for the localized messages that are used in the OVF descriptor.

Another method of using localized resources is to reference external files based on the current location. This method can be used, for example, to display a license text based on the location. The license text is contained in a text file per location. The following example shows how to reference an external plain text file to display a localized license.

```
<Envelope xml:lang="en-US">
   <References>
      <File ovf:id="license-en-US" ovf:href="license-en-US.txt"/>
      <File ovf:id="license-de-DE" ovf:href="license-de-DE.txt"/>
   </References>
    ...
   <VirtualSystem ovf:id="...">
      <EulaSection>
         <Info>Licensing agreement</Info>
         <License ovf:msgid="license">Unused</License>
      </EulaSection>
    ...
   </VirtualSystem>
    ...
   <Strings xml:lang="en-US">
      <Msg ovf:msgid="license" ovf:fileRef="license-en-US">Invalid license</Msg>
   </Strings>
   <Strings xml:lang="de-DE">
      <Msg ovf:msgid="license" ovf:fileRef="license-de-DE">Ihre Lizenz ist nicht
      gültig</Msg>
   </Strings>
</Envelope>
```

The `License` element contains an `ovf:msgid` attribute. In the `Strings` sections, the `ovf:msgid` for the different locations is linked to a file reference by using the `ovf:fileRef` attribute. The `ovf:fileRef` attribute has a corresponding entry in the `References` section of the OVF descriptor. The entry in the `References` section resolves to an external text file that contains the license text.

5.3 Extensibility

The OVF specification allows custom metadata 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.

A design goal of the OVF specification is to ensure backward and forward compatibility. For forward compatibility, this means that an OVF descriptor using features of a later specification (or custom extensions) can be understood by an OVF consumer that either i) is written to an earlier version of the specification, or ii) has no knowledge of the particular extensions. The 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.

5.3.1 Substitution group

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 is optional.

Additional sections can be inserted into the OVF descriptor by defining new members of the `ovf:Section` substitution group. This means the new section extends the base schema for a `Section` element. New sections can be used to define metadata that is not related to the existing sections defined in the OVF specification. The new `Section` has an `<Info>` element that is used to display information to the consumer regarding the section in case the deployment function does not understand the section.

The example shows the addition of a new `Section <ns:BuildInformationSection>`. The `Section` uses the namespace `ns`. The namespace is referenced in a parent element e.g., in the `<Envelope>` element: `<Envelope xmlns="http://schemas.dmtf.org/ovf/envelope/2" xmlns:ns=”http://acme.org/ovf/extension/ns”>`. As required by the `ovf:Section substitutionGroup`, the new section contains an `<Info>` element. The elements `BuildNumber`, `BuildDate`, `BuildSystem` are new elements. The elements are defined in the referenced namespace schema. The `ovf:required` attribute is set to “false” to indicate that the deployment function warns but does not fail if it cannot implement the section.

Example of adding a new section:

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

The XSD schema for the additional section in the example above looks as follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:ns=http://acme.org/ovf/extension/ns
  xmlns:ovf=http://schemas.dmtf.org/ovf/envelope/2
  xmlns:xs=http://www.w3.org/2001/XMLSchema
  targetNamespace=http://acme.org/ovf/extension/ns
  elementFormDefault="qualified"
  attributeFormDefault="qualified">
<xs:import namespace=http://schemas.dmtf.org/ovf/envelope/2
  schemaLocation="dsp8023.xsd"/>
    <xs:element name="BuildInformationSection" type="ns:BuildInformationSection_Type"
    substitutionGroup="ovf:Section">
      <xs:annotation>
        <xs:documentation>Element substitutable for Section since
        BuildInformationSection_Type is a derivation of Section_Type
        </xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:complexType name="BuildInformationSection_Type">
    <xs:annotation>
      <xs:documentation>An ACME specific section.</xs:documentation>
    </xs:annotation>
      <xs:complexContent>
        <xs:extension base="ovf:Section_Type">
          <xs:sequence>
          <xs:element name="BuildNumber" maxOccurs="unbounded">
            <xs:complexType>
              <xs:anyAttribute namespace="##any" processContents="lax"/>
            </xs:complexType>
          </xs:element>
          <xs:element name="BuildDate" maxOccurs="unbounded">
            <xs:complexType>
              <xs:anyAttribute namespace="##any" processContents="lax"/>
            </xs:complexType>
          </xs:element>
          <xs:element name="BuildSystem" maxOccurs="unbounded">
            <xs:complexType>
              <xs:anyAttribute namespace="##any" processContents="lax"/>
            </xs:complexType>
          </xs:element>
          </xs:sequence>
          <xs:anyAttribute namespace="##any" processContents="lax"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
</xs:schema>
```

The schema defines a `BuildInformationSection` substitution group for the `ovf:Section` section. The `BuildInformationSection` substitution group is of the `BuildInformationSection_Type` type. `BuildInformationSection_Type` type defines `ovf:Section_Type` as a base type and extends the `ovf:Section_Type` by the `BuildNumber`, `BuildDate` and `BuildSystem` elements.

5.3.2 Elements

New elements within existing sections can be added at the end of the section. The `Envelope`, `VirtualSystem`, `VirtualSystemCollection`, `Content` and `Strings` sections do not support the addition of elements at the end of the section. The used namespace needs to be referenced in a parent element and must be different from the OVF namespace. Additional elements can be used to extend the information given for a particular section in the OVF descriptor.

An illustration of extending an existing section is given below.

```
<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>
```

The example shows an additional element in the `Annotation` section. The element extends the `Annotation` section with information regarding the `Author` of the descriptor. The new element belongs to the `ns` namespace.

5.3.3 Attributes

A third option of extending an OVF descriptor with additional information is to add custom attributes into existing elements. These attributes can be used to extend the information given by an existing element.

```
<!—— Optional custom attribute example -->
    <Network ovf:name="VM network" ns:desiredCapacity="1 Gbit/s">
        <Description>The main network for VMs</Description>
    </Network>
```

The example above shows the addition of a `desiredCapacity` attribute for the `Network` element. The new attribute is defined in the `ns` namespace.

See ANNEX E for more detailed examples of OVF document extensions.

5.4 Conformance

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

- OVF descriptor only contains metadata defined in the OVF specification, i.e., no custom extensions are present.
  Conformance Level: 1.
- OVF descriptor contains metadata with custom extensions, but all such extensions are optional.
  Conformance Level: 2.
- OVF descriptor contains metadata with custom extensions, and at least one such extension is required.
  Conformance Level: 3.

The use of conformance level 3 limits portability, which means that the OVF package might not be deployed on any other virtualization platform other than the one supporting the custom extensions.

5.5 Virtual hardware description

The hardware description shown below is very general. In particular, it 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 is appropriate. More specifically, it can be assumed that if the specification is generic, the appliance undertakes discovery of the devices present, and loads relevant drivers. In this case, it is 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 deployed on a virtualization platform that does not offer the same hardware devices and/or categories of devices that are required by the guest software that is included in the appliance, installation failures that are nontrivial and nonobvious can occur. The risk is that it fails to install and/or boot, and that the user is not able to debug the problem. With this situation 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 (because the specific devices required are listed), but this specification limits the number of systems that the appliance may be able install on and/or boot.

Consider that simplicity, robustness, and predictability of installation are key reasons that ISVs are moving to the virtual appliance model. 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 software used in the virtual machines and the range of virtual hardware the guest software supports.

Ideally, the appliance vendor creates 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-to-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 mode 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 reboots (for example, via floppy or CD drives attached to the guest on first boot). In addition, for sysprepped Windows VMs, that need only re-installation and customization with naming etc., the reboot technique allows naming and tailoring of the image to be achieved without consumer intervention.

The following example illustrates multiple virtual hardware profiles for different virtualization platforms specified in the same descriptor.

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

This type of profile 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 deployment function chooses the latest or most capable feature set that is available on the local platform.

The example below shows how a specific type of virtual hardware can be defined. Multiple options for the `rasd:ResourceSubType` can be separated by a single space character. The deployment function can then choose the virtual hardware type to instantiate.

If multiple resource subtypes are specified in the OVF package, the deployment function selects an appropriate value to populate the CIM_ResourceAllocationSettingData class for the managed environment. If more than one appropriate value exists the deployment function selects any of the appropriate values. If the deployment function does not find an appropriate value the deployment fails.

```
<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>
```

5.6 Example descriptors

The following examples have been provided as complete examples of an OVF descriptor. These examples pass XML validation.

ANNEX A illustrates an OVF descriptor for a single virtual system.

ANNEX B illustrates a multiple virtual system OVF descriptor.

ANNEX C illustrates an OVF descriptor for a single virtual system with multiple applications contained in it; i.e., a LAMP stack.

ANNEX D illustrates an OVF descriptor for a multiple virtual system with multiple applications contained in it, i.e., a LAMP stack with two virtual systems.

6 Deploying an OVF package

6.1 Deployment

Deployment transforms the virtual machines in an OVF package into the run-time 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 process includes assigning and configuring the networks (physical and virtual) too which the virtual machines are 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 shows, or prompts for, the following information:

- Show information about the OVF package (from the `ProductSection`), and ask the user to accept the licensing agreement, or address 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 the physical networks to which the logical networks in the OVF package are connected.
- 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 started successfully 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. After 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.

6.2 OVF environment descriptor

The OVF environment descriptor is an XML document that describes metadata about the software installed on the virtual disks. The OVF specification defines the common sections used for deploying software, 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.

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 is specified in the OVF descriptor by using the `ProductSection` metadata, and is 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 multitiered 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, because no standard cross-vendor para-virtualized device exists for communicating between the guest software 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:

- a standard *protocol* that specifies what information is available and in what format it is available
- a *transport* that specifies how the information is obtained

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

6.3 Resource configuration options during deployment

The OVF package has the ability to include resource configuration options for a virtual appliance. This makes it easy for the package consumer to get an initial setup without having to make individual resource decisions based on the intended use. The `Description` and `Label` elements provide a human-readable list of resource configurations, for example:

- Software evaluation setup
- 10-100 person workgroup setup
- 100-1000 person workgroup setup
- Large enterprise workgroup setup

The deployment function prompts for selection of a configuration. The list of configurations described above is expected to be used for a suitable initial resource configuration.

Example list of configurations:

```
<DeploymentOptionSection>
    <Configuration ovf:id="eval">
        <Label>Software Evaluation</Label>
        <Description>Software evaluation setup</Description>
    </Configuration>
    <Configuration ovf:id="small" ovf:default="yes">
        <Label>Small</Label>
        <Description>10-100 person workgroup setup</Description>
    </Configuration>
    <Configuration ovf:id="medium">
        <Label>Medium</Label>
        <Description>100-1000 person workgroup setup</Description>
    </Configuration>
    <Configuration ovf:id="large">
        <Label>Large</Label>
        <Description>Large enterprise workgroup setup</Description>
    </Configuration>
</DeploymentOptionSection>
```

The following snippet of an OVF descriptor illustrates the configuration option use for a resource requirement. In this case, if the consumer chose 'eval', the resource allocation data used is that in the `Item ovf:configuration="eval"`.

Resource requirement example:

```
<ResourceAllocationSection>
   <Info>Defines reservations for CPU and memory</Info>
   <Item>
        ... default configuration ...
   </Item>
   <Item ovf:configuration="eval">
        ... replaces the default configuration if the "eval" configuration if selected
...
   </Item>
</ResourceAllocationSection>
```

The following snippet of an OVF descriptor illustrates the configuration option use for a `VirtualHardwareSection`. In this case, if the consumer chose “large”, the resource allocation data used is that in the `Item ovf:configuration=”large”`.

`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:Limit>1100</rasd:Reservation>
      <rasd:ResourceType>3</rasd:ResourceType>
      <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
  </Item>
  ...
  <Item ovf:configuration="large">
      <rasd:AllocationUnits>hertz * 10^6</rasd:AllocationUnits>
      <rasd:ElementName>1 CPU and 800 MHz reservation</rasd:ElementName>
      <rasd:InstanceID>1</rasd:InstanceID>
      <rasd:Reservation>800</rasd:Reservation>
      <rasd:ResourceType>3</rasd:ResourceType>
      <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
  </Item>
</VirtualHardwareSection>
```

6.4 Product customization during deployment using Property elements

The OVF descriptor can contain a description of the guest software that includes information about customization provided through the OVF environment. This information is provided by the use of `Property` elements in the `ProductSection` of the OVF descriptor.

Each `Property` element has five possible attributes: `ovf:key, ovf:type, ovf:qualifiers, ovf:value,` and `ovf:userConfigurable.`

The `ovf:key` attribute is a unique identifier for the `Property` element.

The `ovf:type` attribute indicates the type of value the `Property` element contains.

The `ovf:qualifiers` attribute specifies additional information regarding the `ovf:type` attribute so that CIM value maps can be used.

The `ovf:value` attribute is used to provide a value for a `Property` element.

The `ovf:userConfigurable` attribute determines whether the value provided is a default value and changeable at deployment or is not changeable.

An example of the use of `Property` elements follows.

```
<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>
  <Property ovf:key="Gateway" ovf:type="string" ovf:value=”192.168.0.1”
    ovf:userConfigurable="false" >
    <Description>Gateway address to be used</Description>
  </Property>
  <Property ovf:key=" SoftwareResourceType" ovf:type="uint16"
    ovf:qualifiers=”uint16,uint16,uint16,uint16,uint16,uint16,uint16,uint16,uint16,
      uint16,”
    ovf:value="Unknown", "Buffer", "Queue", "Protocol Endpoint", "Remote Interface",
      "Pool", "Cache", "File", "Database",
    ovf:userConfigurable="false" >
    <Description>Value Map example based on SoftwareResourceType property in
      CIM_SoftwareResource class</Description>
  </Property>
</ProductSection>
```

In the `CIM_SoftwareResource` class, there is a property `SoftwareResourceType` that is shown in the snippet from the CIM Schema below.

```
      [Description (
          "The type of the software resource. Although the behavior "
          "of the different software resource types is modeled "
          "similarly, different names for resources transferring "
          "data over time or/and space have been established. "
          "SoftwareResourceType conveys their original, most common "
          "name. \n"
       ValueMap { "0", "2", "3", "4", "5", "6", "7", "8", "9",
          "10..32767", "32768..65535" },
       Values { "Unknown", "Buffer", "Queue", "Protocol Endpoint",
          "Remote Interface", "Pool", "Cache", "File", "Database",
          "DMTF Reserved", "Vendor Reserved" }]
   uint16 SoftwareResourceType;
```

The `ovf:type` corresponds to the index of the `ValueMap` so it is `uint16`.

ANNEX D contains a detailed example of customization of a complex multitiered application.

7 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 metadata enables a robust and user-friendly experience when installing a virtual appliance. In particular, the metadata can be used by the management infrastructure to confidently decide whether a particular VM described in an OVF can be installed or whether it is rejected, and potentially to guide appropriate conversions and localizations to make it useable in the specific execution context that 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 installability 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 are able to import and 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.
- 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 requires interaction with a floppy disk controller and therefore it is not 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. Because the OVF standard evolves independently of virtualization products, at any point an OVF might be unsupportable on a virtualization platform that predates that OVF specification.

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

- **Portability class 1**. Runs on multiple families of virtual hardware. For example, the appliance could be used 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 guest software, during which the guest software 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 guest software instances can be re-installed, re-named, and re-personalized on multiple hardware platforms, including virtual hardware.
- **Portability class 2**. Runs on a specific family of virtual hardware. This is typically due to lack of driver support by the installed guest software.
- **Portability class 3**. Only runs on a particular virtualization product and/or CPU architecture and/or virtual hardware selection. This is typically 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 a state ties the OVF to a very specific virtualization and hardware platform.

For use within an organization, class 2 or class 3 compatibility may be good enough because 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 creates an OVF with class 3 or class 2 portability (tied to a specific set of virtual hardware); however, it is easy to extend the metaphor to support the export of class 1 portability, for example through the use of utilities such as “sysprep” for Windows.

For commercial appliances independently created and distributed by ISVs, class 1 portability is desirable. Indeed, class 1 portability ensures that the appliance is readily available for the broadest possible customer base both for evaluation and production. Toolkits are used to create certified “known good” class 1 packages of the appliance for broad distribution and installation on multiple virtual platforms, or class 2 portability 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 class 1 through class 3 portability. For class 1 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.

ANNEX A
(informative)

Single virtual system example

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 five sections.

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 services are 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="adminEmail" ovf:type="string">
                <Description>Email address of administrator</Description>
            </Property>
            <Property ovf:key="appIp" 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>
```

ANNEX B
(informative)

Multitiered pet store example

The Pet Store OVF descriptor demonstrates several advanced OVF concepts:

- Multi-VM packages - use of the VirtualMachineCollection entity subtype.
- Composite service organization - use of a nested VirtualMachineCollection entity subtype.
- Propagation of a 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 `ProductSection` elements for providing information about the installed software in an individual virtual machine.

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

B.1 Architecture and packaging

The Pet Store OVF package consists of three virtual systems (WebTier, DB1, and DB2) and two virtual system collections (Pet Store and DBTier). Figure B-1 shows the structure of the OVF package as well as the properties and startup order of the virtual machines.

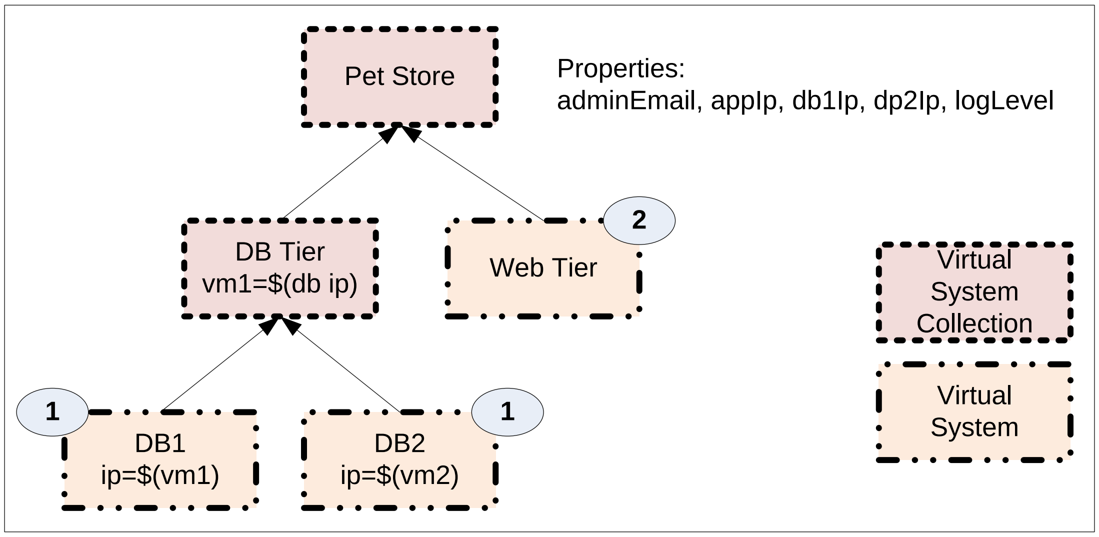


Figure B-1 – Pet Store OVF package

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 clauses.

B.2 Properties

The Pet Store service has five user-configurable properties. These are the key control parameters for the service that need 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 described in the following tables.

Table B-1 illustrates the properties for the Web Guest Software:

Table B-1 – Web tier configuration

Property	**Description**
appIp	IP address of the Web Server
dbIp	IP address of the database server to which to connect
adminEmail	Email address for support
logLevel	Logging level

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

Table B-2 illustrates the properties for the Database Guest Software:

Table B-2 – Database tier configuration

Property	**Description**
Ip	IP address of the virtual machine
primaryAtBoot	Whether the instance acts as the primary or secondary when booting
ip2	IP address of the twin database VM that acts as the hot-spare or primary
log	Logging level (called log here)

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 "com.mydb.db" as the unique name for all the properties. This string 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 for the Petstore virtual system are not available to the DB1 virtual system. 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 boot up into different roles even though they are, initially, booting from the exact same software image. The property named *com.mydb.db.primaryAtBoot* is specified with a fixed, non-user configurable value but is different for the two images. The software inspects this value at boot time and customizes its operation accordingly.

B.3 Disk layout

The Pet Store 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 built on top of the same base OS distribution, we can encode this configuration in the OVF descriptor as shown in Figure B-2.

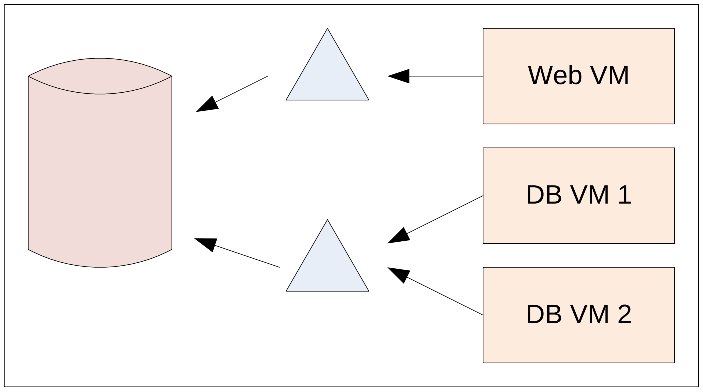


Figure B-2 – Pet Store virtual disk layout

Thus, while the package contains three distinct virtual machines, the total download size is 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 gets independent disks. Thus, a virtualization platform could install the above package as three VMs with three independent flat disks, or it could chose to replicate the above organization, or something else, as long as each virtual machine sees a disk with the content described on initial boot and that changes written by one virtual machine do not affect the others.

B.4 Pet Store 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 services
                are 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 multitier 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>
                        <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>
```

B.5 Complete OVF environment

The following example lists the OVF environments that are seen by the WebTier and DB1 virtual machines. (DB2 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>
```

ANNEX C (informative)

Single virtual system LAMP stack example

In this example, we provide two concrete examples of how an OVF descriptor for a LAMP virtual appliance could look. We show both a single-VM LAMP virtual appliance and a multi-VM LAMP virtual appliance. LAMP is an abbreviation for a service built by 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; for example, there are no EULA sections.

C.1 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 the 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 are exposed to the deployer vary from vendor to vendor and service to service. In the example descriptors, the following set of properties described in Table C-1 are used for the four different LAMP components:

Table C-1 – LAMP configuration

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	

Product	Property	Description
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 configurations 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 are 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>
```

C.2 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 is 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 are
                    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>
```

ANNEX D (informative)

Multiple virtual system LAMP stack example

This example is what an OVF descriptor for a LAMP virtual appliance could look like in 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.

D.1 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 the process of how a service is implemented transparent to the user. In particular, the deployment experience when a user is installing a single-VM or a two-tier LAMP appliance is very similar. The only visible difference is that the user needs to supply two IP addresses and two DNS host names.

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

- All the user-configurable parameters are put in the `VirtualSystemCollection` entity. The `ProductSection` elements 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. Because 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 is 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 ovf:class="org.mylamp">
            <Info>Product information for the service</Info>
            <Product>My Lamp Service</Product>
            <Version>1.0</Version>
            <FullVersion>1.0.0</FullVersion>
        </ProductSection>
        <ProductSection ovf:class="org.linuxdist">
            <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 are
                    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>
```

ANNEX E (informative) Extensibility example

The OVF specification allows custom metadata to be added to OVF descriptors in several ways:

- New section elements can 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 can 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 metadata 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.

E.1 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 >
```

E.2 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.

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 metadata 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.

ANNEX F (informative) Change log

Version	Date	Description
1.0.0	2009-02-17	
1.0.1	2011-10-20	Release for DMTF errata publication
2.0.0	2014-04-24	Release for DMTF publication as Informational