Document Number: DSP0265

Date: 2012-10-25

Version: 1.0.0a

Profile to Enable Automated Deployment of OVF Packages

Information for Work-in-Progress version:

IMPORTANT: This document is not a standard. It does not necessarily reflect the views of the DMTF or all of its members. Because this document is a Work in Progress, it may still change, perhaps profoundly. This document is available for public review and comment until the stated expiration date.

It expires on: 2013-03-31

Provide any comments through the DMTF Feedback Portal:
http://www.dmtf.org/standards/feedback

Document Type: Specification

Document Status: Work in Progress

Document Language: en-US

CONTENTS

Figures

Tables

Foreword

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

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

Acknowledgments

The DMTF acknowledges the following individuals for their contributions to this document:
Editors:
Richard Landau - DMTF Fellow
Lawrence Lamers -VMware Inc.

Contributors
Hemal Shah - Broadcom
John Crandall - Brocade Communications Systems
Marvin Waschke - CA Technologies
Shishir Pardikar - Citrix Systems Inc.
Eric Wells - Hitachi, Ltd.
Robert Freund - Hitachi, Ltd.
HengLiang Zhang - Huawei
Jeff Wheeler - Huawei
Abdellatif Touimi - Huawei
Andreas Maier - IBM
Ron Doyle - IBM
John Leung - Intel Corporation
Cheng Wei - Microsoft Corporation
John Parchem - Microsoft Corporation
Monica Martin - Microsoft Corporation
Maurizio Carta - Microsoft Corporation
Narayan Venkat - NetApp
Srinivas Maturi - Oracle
Tatyana Bagerman - Oracle
Miguel PeñalvovTelefónica
Steffen Grarup - VMware Inc.
Bhumip Khasnabish - ZTE Corporation
Junsheng Chu - ZTE Corporation
Ali, Ghazanfar - ZTE Corporation

Introduction

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

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

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

Profile to Enable Automated Deployment of OVF Packages

1 Scope

1.1 Metadata Structure

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

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

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

- This profile applies to a specific application area, namely, the automated deployment of virtual systems contained in OVF packages for business and scientific applications in datacenters.

- This profile states conformance rules for OVF v1.1 Envelope files intended to be applied in this area, That is, if an OVF package is intended to be used in an environment where it will be deployed by automated mechanisms, with little or no human intervention, then the Envelope of the package should conform to this profile.

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

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

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

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

1.2 The Meaning of Automated Deployment

The goal of automated deployment is to minimize the human interaction required at the moment of deployment of a virtual system. Ideally, the act of deploying a virtual machine might be a one-button process: push the button and an instance of a virtual machine comes to life. This may be achievable to greater or lesser degrees in various datacenter environments. The goal of this specification is to increase

the degree of automation that can be achieved by minimizing the intervention of a human system manager at the time of deployment.

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

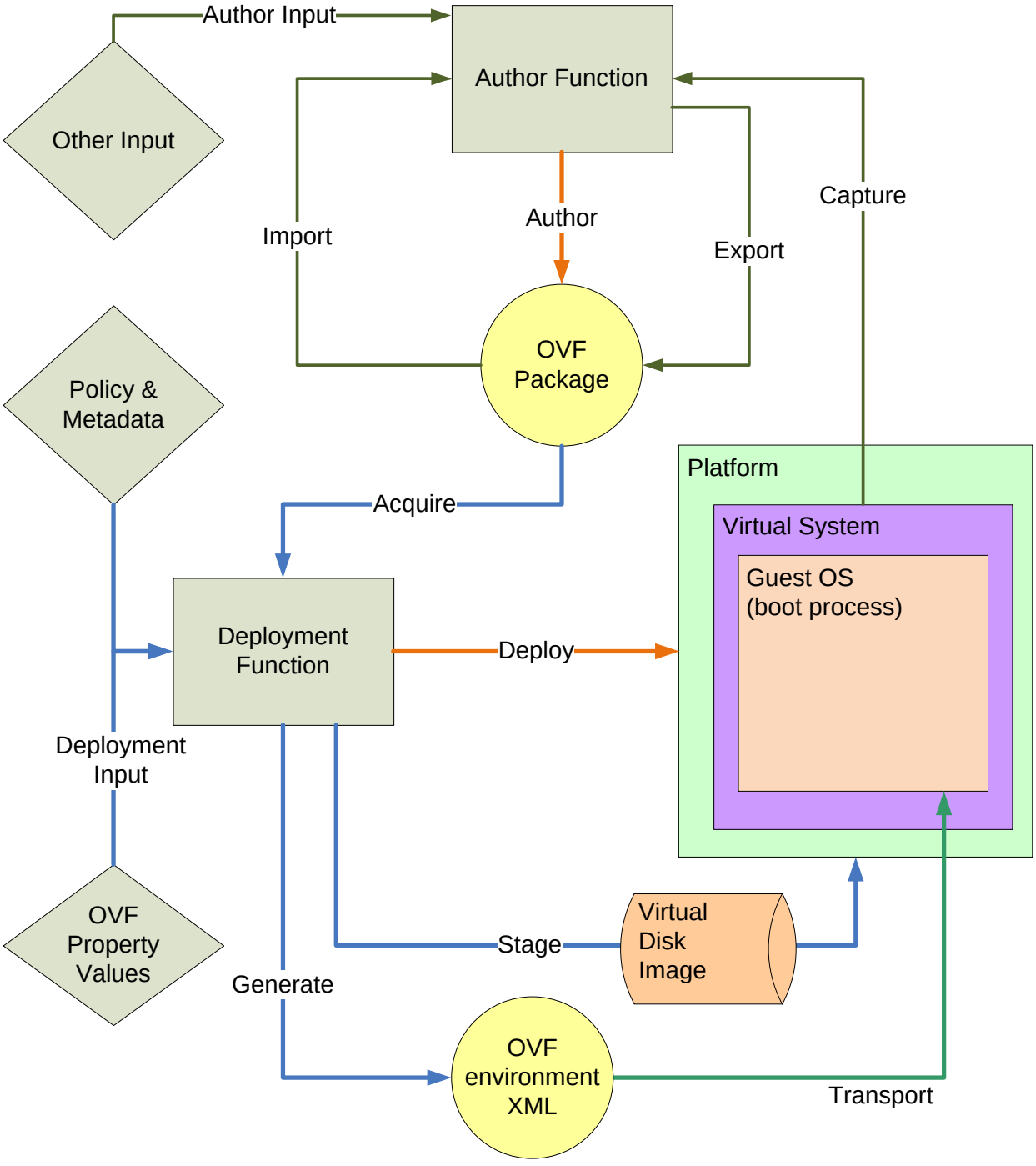
A deployment agent can find a suitable virtual machine slot for a system only if it knows in advance about the type of system required by the virtual machine, processors needed, memory, peripherals, and such. The OVF Envelope of a virtual machine can supply this type of information. Additionally, a deployment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

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

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

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

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



210

211 **Figure 1 – OVF Package Lifecycle**

2 Normative References

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

2.1 Approved References

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

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

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

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

DMTF CIM Schema 2.34.

2.2 References under Development

2.3 Other References

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

3 Terms and Definitions

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

3.1
can
used for statements of possibility and capability, whether material, physical, or causal

3.2
cannot
used for statements of possibility and capability, whether material, physical or causal

3.3
conditional
indicates requirements to be followed strictly in order to conform to the document when the specified conditions are met

3.4
Interoperability
ISO/TR 16056-1:2004, 3.42
The ability of two or more systems (computers, communication devices, networks, software, and other information technology components) to interact with one another and exchange information according to a prescribed method in order to achieve predictable results.

3.5
mandatory
indicates requirements to be followed strictly in order to conform to the document and from which no deviation is permitted

3.6
may
indicates a course of action permissible within the limits of the document

3.7
need not
indicates a course of action permissible within the limits of the document

3.8
optional
indicates a course of action permissible within the limits of the document

3.9
OVF Portability
The ability to export a virtual system from one author function to be imported into another author function.

3.10
referencing profile
indicates a profile that owns the definition of this class and can include a reference to this profile in its "Related Profiles" table

3.11
shall
indicates requirements to be followed strictly in order to conform to the document and from which no deviation is permitted

3.12
shall not
indicates requirements to be followed strictly in order to conform to the document and from which no deviation is permitted

3.13
should
indicates that among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others, or that a certain course of action is preferred but not necessarily required

3.14
should not
indicates that a certain possibility or course of action is deprecated but not prohibited

3.15
unspecified
indicates that this profile does not define any constraints for the referenced CIM element or operation

3.16
Workload
A defined set of operations for the deployed ovf package to perform that validates the functionality and that is visible in some fashion.

4 Abbreviated Terms and Document Conventions

4.1 Abbreviated Terms

The following abbreviations are used in this document.

4.1
EPASD
CIM_EthernetPortAllocationSettingData that represents settings specifically related to allocation of an Ethernet port resource.

4.2
LAMP
A Linux, Apache, MySQL and PHP (aka LAMP) OVF package that illustrates a multiple virtual machine deployment.

4.3
OVF
Open Virtualization Format

4.4
RASD
CIM_ResourceAllocationSettingData that represents settings specifically related to an allocated resource.

4.5
VM
Virtual Machine

4.6
DA
Deployment Agent

4.2 Document Conventions

4.2.1 Typographical Conventions

5 Synopsis

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

Version: 1.0.0

Organization: SVPC OVF Working Group

CIM Schema Version: 2.29

Central Class: n/a

Scoping Class: n/a

OVF Specification Version: DSP0243 v1.1

6 Description (informative)

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

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

6.1 Problem Area

This profile is targeted at deployment of virtual systems in medium businesses to large enterprises. The virtual systems will be business, academic, or scientific appliances working in a network environment. This profile is not targeted at small, one-on-one applications that do not use a network as a fundamental part of the application.

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

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

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

6.2 Different Kinds of Incomplete Information

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

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

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

6.3 Information Needed for Deployment

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

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

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

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

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

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

7 Implementation (informative)

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

8 Methods (informative)

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

9 Use Cases (informative)

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

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

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

In these use cases, several different types of information may be needed by the deployment agent (DA) depending on the configuration of VMs. To enable automated deployment, this information must be available to the DA in advance of the request to deploy one or more copies of a VM.

- Information relating to the structure of the VM to be activated: the operating system, instruction set, memory, CPU, disk, and peripheral requirements, and so forth. The DA must choose a suitable virtual hardware system and provide the required virtual hardware resources for the VM.

- Information that will make each system unique when multiple copies of the VM are being run in the same naming domain, e.g., scaled-out in a load balanced set: system name, IP address for management, other network addresses that are known globally, etc. To launch multiple copies of a VM, the DA must be able to structure the copies to avoid naming and addressing conflicts when all are active. If multiple copies are hidden behind a load balancer, then the DA needs to assign predictable IP addresses (and names, if used) to the load balancer and its subordinate VMs. The DA needs to understand in advance what network connections will be used by VMs for what purposes.

- Information that enables multiple VMs to cooperate as a single application service: e.g., names and addresses of partner and sibling components for a tiered application.

    - A web server VM (or multiple front-end web server VMs) need to have access to the application VM (or VMs) for the business logic of the application; to authentication and authorization servers or databases; and to a store of web pages that may be managed by a separate VM.

    - An application VM needs to have access to the database of information being served and to the store of application programs.

    - The addresses of the several components are not known in advance to the VMs, but are supplied by the DA at the time of activation. To enable automated deployment, the DA must know in advance the needs of each VM, and must have control over delivering addresses to the VMs. Again, to automate deployment with minimal human intervention, the DA needs to understand in advance what network connections will be used by VMs for what purposes.

9.1 Deploy Copies of Virtual System Containing Guest OS Only

The user wants to deploy copies of a virtual system as development, training, and test platforms. The virtual system contains only an operating system built to a standard configuration.

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

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

9.2 Deploy LAMP Server in a Single Virtual System

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

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

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

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

9.3 Deploy LAMP Server in Separate Virtual Systems

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

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

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

10 OVF Requirements

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

10.1 Mandatory Sections

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

Table 1 – OVF Sections Required

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

10.2 Mandatory Elements

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

Table 2 – OVF Elements Required

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

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

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

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

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

503 **Table 3 – OVF RASD Elements Required**

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

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

506 **Table 4 – OVF RASD Element Properties Required**

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

507

10.3 Naming Properties

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

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

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

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

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

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

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

10.4 Conditionally Mandatory Complete Property Groups

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

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

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

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

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

552 **Table 5 – Groups of Related Properties Network Address**

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

553 Note:
554 • Property keys and values are case sensitive.
555 • Property keys that end with a dot and number could be a list.
556 • Some keys may depend on others to have a meaning, e.g., network.ipaddr_IPv4 is meaningful if
557 network.bootproto is not 'dhcp'.

558 **Table 6 – Groups of Related Properties for Email Contact**

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

559 Email To address
560 string ToAddress
561 String designating the intended recipient or recipients of the message. The string shall contain a
562 mailto URI as specified in RFC6068. This URI may contain multiple recipients and CC and BCC
563 options, within the limits specified by the RFC.

564 SMTP service address including port
565 string SMTPServiceAddress
566 String specifying the IP address and port of an SMTP service to be used to transmit the email. If the
567 string does not specify the port, then the default port specified in RFC6409 shall be used.

568 SMTP login name
569 string CIM_Account.Name
570 For an SMTP service that is listed in the CIM_Account.Host array of a CIM_Account, this shall be the
571 name of an account that is privileged to send email through the service.

572 SMTP login password
573 string CIM_Account.UserPassword
574 For a CIM_Account on an SMTP service that is listed in the CIM_Account.Host array of a
575 CIM_Account, this shall be the password associated with the CIM_Account.Name.

576 SMTP login requires SPA authentication
577 boolean SMTPIsSPAAuthenticationRequired
578 Boolean specifying whether SPA (Secure Password Authentication) is required to access the SMTP
579 service.

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

10.5 Properties for Multiple Instances of Network Interfaces

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

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

Specifically, in the ovf:Envelope,

- Property declarations that reference multiple instances of an Ethernet interface within a ProductSection shall specify the corresponding ovf:Item in the ovf:VirtualHardwareSection; the Property declaration shall include a dsp265:rasdinstanceid attribute the value of which is the rasd:InstanceID value of the ovf:Item.

- ovf:Item declarations in the ovf:VirtualHardwareSection shall include a rasd:Connection item specifying the network to which the interface is to be attached.

Example:

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

```
<Envelope xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1"
xmlns:dsp265="http://schemas.dmtf.org/ovf/dsp0265/1">

    <!--
    Request for information for two ethernet interfaces.

    Note that the instance ids point to the corresponding RASD elements for the
    interfaces; the RASD element rasd:Connection specifies which physical or
    logical network the interface is attached to.  The deployment agent needs
    to be able to distinguish the several (virtual) interfaces in order to
    assign them addresses on their specific networks.  To accomplish this,
    a new attribute, dsp265:rasdinstanceid, contains the RASD InstanceID string
    in both the Envelope request and the Environment reply.
    -->

    <ProductSection>
        <Info></Info>
        <Product></Product>
        <Vendor></Vendor>
        <Version></Version>
        <Property ovf:key="CIM_StaticIPAssignmentSettingData.IPv4Address"
ovf:type="string" dsp265:rasdinstanceid="6">
```

```
            <Label>Ethernet on VM Network</Label>
            <Description>IPv4 addr of this VM on the VM Network for normal external
and internal data</Description>
        </Property>
        <Property ovf:key="CIM_StaticIPAssignmentSettingData.IPv4Address"
ovf:type="string" rasdinstanceid="7">
            <Label>Ethernet on Admin Network</Label>
            <Description>IPv4 addr of this VM for private traffic on the
administrative network</Description>
        </Property>
    </ProductSection>

</Envelope>
```

And the Environment constructed by the Deployment Agent may contain responses such as the following.

```
<Environment xmlns:ovfenvir="http://schemas.dmtf.org/ovf/environment/1"
xmlns:dsp265="http://schemas.dmtf.org/ovf/dsp0265/1">

    <!--
    Environment response to requests in the Envelope.
    -->

    <ProductSection>
        <Property ovfenvir:key="CIM_StaticIPAssignmentSettingData.IPv4Address"
ovfenvir:type="string" dsp265:rasdinstanceid="6" ovfenvir:value="10.0.1.1">
        </Property>
        <Property ovfenvir:key="CIM_StaticIPAssignmentSettingData.IPv4Address"
ovfenvir:type="string" dsp265:rasdinstanceid="7" ovfenvir:value="10.99.1.1">
        </Property>

    </ProductSection>


</Environment>
```

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

11 Conformance

11.1 Citation in OVF Envelope Element

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

```
<ovfprofiles:ProfileSupported>
  <ovfprofiles:ProfileName>Profile to Enable Automated Deployment of OVF
Packages</ovfprofiles:ProfileName>
  <ovfprofiles:ProfileVersion>1.0</ovfprofiles:ProfileVersion>
</ovfprofiles:ProfileSupported>
```

11.2 XML Namespace

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

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

ANNEX A
(informative)

Use Cases for Future Consideration

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

A.1 Deploy General Tiered Application

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

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

A.2 Deploy Scaled-out Copies of Virtual System

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

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

ANNEX B
(informative)

Change Log

Version	Date	Author	Description
1.0.0a	2012-10-25	Larry Lamers	Release as DMTF Work In Progress (wgv 0.7.2)

Bibliography