Open Virtualization Format Specification

Information for work in progress version:
"IMPORTANT: This specification 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, this specification may still change, perhaps profoundly. This document is available for public review and comment until the stated expiration date."

It expires on: 2012-12-15
Target version for final status 2.0.0c
Provide any comments through the DMTF Feedback Portal: http://www.dmtf.org/standards/feedback/

Document Type: Specification
Document Status: Work in Progress
Document Language: E
Copyright notice

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

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

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patent owner or claimant, and shall have no liability or responsibility for costs or losses incurred if a standard is withdrawn or modified after publication, and shall be indemnified and held harmless by any party implementing the standard from any and all claims of infringement by a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

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

41 Foreword .................................................................................................................. 6
42 Introduction ............................................................................................................... 7
43 1 Scope ..................................................................................................................... 8
44 2 Normative References .......................................................................................... 8
45 3 Terms and Definitions ........................................................................................... 9
46 4 Symbols and Abbreviated Terms .......................................................................... 11
47 5 OVF Packages ....................................................................................................... 11
48 5.1 OVF Package Structure ....................................................................................... 11
49 5.2 Virtual Disk Formats .......................................................................................... 13
50 5.3 Distribution as a Single File ............................................................................... 13
51 5.4 Distribution as a Set of Files ............................................................................ 14
52 6 OVF Descriptor ..................................................................................................... 14
53 7 Envelope Element ................................................................................................. 15
54 7.1 File References ................................................................................................... 16
55 7.2 Content Element ............................................................................................... 18
56 7.3 Extensibility ....................................................................................................... 18
57 7.4 Conformance ...................................................................................................... 20
58 8 Virtual Hardware Description ................................................................................ 20
59 8.1 VirtualHardwareSection .................................................................................... 20
60 8.2 Extensibility ....................................................................................................... 22
61 8.3 Virtual Hardware Elements ............................................................................... 23
62 8.4 Ranges on Elements .......................................................................................... 25
63 9 Core Metadata Sections in version 2 ..................................................................... 27
64 9.1 DiskSection ....................................................................................................... 28
65 9.2 NetworkSection ............................................................................................... 29
66 9.3 ResourceAllocationSection .............................................................................. 30
67 9.4 AnnotationSection ............................................................................................. 31
68 9.5 ProductSection .................................................................................................. 32
69 9.6 EulaSection ....................................................................................................... 35
70 9.7 StartupSection .................................................................................................... 35
71 9.8 DeploymentOptionSection ................................................................................ 36
72 9.9 OperatingSystemSection ................................................................................... 39
73 9.10 InstallSection ................................................................................................... 39
74 9.11 EnvironmentFilesSection ................................................................................ 39
75 9.12 BootDeviceSection .......................................................................................... 40
76 9.13 SharedDiskSection .......................................................................................... 41
77 9.14 ScaleOutSection .............................................................................................. 42
78 9.15 PlacementGroupSection and PlacementSection ............................................ 44
79 9.16 Encryption Section .......................................................................................... 45
80 10 Internationalization ............................................................................................. 47
81 10.1 Internal Resource Bundles ............................................................................... 48
82 10.2 External Resource Bundles ............................................................................. 48
83 10.3 Message Content in External File ..................................................................... 49
84 11 OVF Environment ............................................................................................... 50
85 11.1 Environment Document ................................................................................... 50
86 11.2 Transport .......................................................................................................... 51
87 ANNEX A (informative) Symbols and Conventions .............................................. 53
88 ANNEX B (informative) Change Log ...................................................................... 54
89 ANNEX C (normative) OVF XSD ........................................................................... 55
90 ANNEX D (informative) OVF Mime Type Registration Template ......................... 56
Bibliography ........................................................................................................................................... 58
Tables

94 Table 1 – XML Namespace Prefixes ................................................................. 15
95 Table 2 – Actions for Child Elements with ovf:required Attribute .................... 23
96 Table 3 – HostResource Element .................................................................. 24
97 Table 4 – Elements for Virtual Devices and Controllers ................................. 25
98 Table 5 – Core Metadata Sections in version 2 .................................................. 27
99 Table 6 – Property Types .............................................................................. 34
100 Table 7 – Property Qualifiers .................................................................... 35
101 Table 9 – Core Sections ............................................................................. 51
The Open Virtualization Format Specification (DSP0243) was prepared by the System Virtualization, Partitioning, and Clustering Working Group of the DMTF.

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

Vincent Kowalski, BMC Software
Hemal Shah, Broadcom Corporation
John Crandall, Brocade Communications Systems
Marvin Waschke, CA Technologies
Naveen Joy, Cisco
Steven Neely, Cisco
Shishir Pardikar, Citrix Systems Inc.
Richard Landau, Dell
Jacques Durand, Fujitsu
Derek Coleman, Hewlett-Packard Company
Robert Freund, Hitachi, Ltd.
Fred Maciel, Hitachi, Ltd.
Eric Wells, Hitachi, Ltd.
Abdellatif Touimi, Huawei
Jeff Wheeler, Huawei
Heng Liang Zhang, Huawei
Oliver Benke, IBM
Ron Doyle, IBM
Michael Gering, IBM
Michael Johanssen, IBM
Andreas Maier, IBM
Marc-Arthur Pierre-Louis, IBM
John Leung, Intel Corporation
Nitin Bhat, Microsoft Corporation
Maurizio Carta, Microsoft Corporation
Monica Martin, Microsoft Corporation
John Parchem, Microsoft Corporation
Ed Reed, Microsoft Corporation
Nihar Shah, Microsoft Corporation
Cheng Wei, Microsoft Corporation
Narayan Venkat, NetApp
Tatyana Bagerman, Oracle
Srinivas Maturi, Oracle
Dr. Fermín Galán Márquez, Telefónica
Miguel Ángel Peñalvo, Telefónica
Dr. Fernando de la Iglesia, Telefónica
Álvaro Polo, Telefónica
Steffen Grarup, VMware Inc.
Lawrence Lamers, VMware Inc.
Rene Schmidt, VMware Inc.
Paul Ferdinand, WBEM Solutions
Junshen Chug, ZTE Corporation
Bhumip Khasnabish, ZTE Corporation
Introduction

The Open Virtualization Format (OVF) Specification describes an open, secure, portable, efficient and extensible format for the packaging and distribution of software to be run in virtual machines. The key properties of the format are as follows:

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

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

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

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

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

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

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

- **Open standard**
  OVF has arisen from the collaboration of key vendors in the industry, and it is developed in an accepted industry forum as a future standard for portable virtual machines.

It is not an explicit goal for OVF to be an efficient execution format. A hypervisor is allowed but not required to run software in virtual machines directly out of the Open Virtualization Format.
Open Virtualization Format Specification

1 Scope

The Open Virtualization Format (OVF) Specification describes an open, secure, portable, efficient and extensible format for the packaging and distribution of software to be run in virtual machines.

This version of the specification (2.0) is intended to allow OVF 1.x tools to work with OVF 2.0 descriptors in the following sense:

- Existing OVF 1.x tools should be able to parse OVF 2.0 descriptors.
- Existing OVF 1.x tools should be able to give warnings/errors if dependencies to 2.0 features are required for correct operation.

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.

- DMTF CIM Schema 2.33,
  http://www.dmtf.org/standards/cim
- DMTF DSP0004, Common Information Model (CIM) Infrastructure Specification 2.6,
  http://www.dmtf.org/standards/published_documents/DSP0004_2.6.pdf
- DMTF DSP0230, WS-CIM Mapping Specification 1.0,
  http://www.dmtf.org/standards/published_documents/DSP0230_1.0.pdf
- DMTF DSP1041, Resource Allocation Profile (RAP) 1.1,
- DMTF DSP1043, Allocation Capabilities Profile (ACP) 1.0,
  http://www.dmtf.org/standards/published_documents/DSP1043_1.0.pdf
- DMTF DSP8023, Open Virtualization Format (OVF) 2.0 XML Schema,
  http://schemas.dmtf.org/ovf/envelope/2/dsp8023_2.0.0.xsd
- IETF RFC1738, T. Berners-Lee, Uniform Resource Locators (URL), December 1994,
- IETF RFC1952, P. Deutsch, GZIP file format specification version 4.3, May 1996,
- IETF Standard 68, Augmented BNF for Syntax Specifications: ABNF,
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 to conform to the document when the specified conditions are met

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

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

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

3.7 optional
indicates a course of action permissible within the limits of the document
shall indicates requirements to be followed strictly to conform to the document and from which no deviation is permitted.

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

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.

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

appliance see virtual appliance

deployment platform the product that installs an OVF package

guest software the software, stored on the virtual disks, that runs when a virtual machine is powered on.

The guest is typically an operating system and some user-level applications and services.

OVF package OVF XML descriptor file accompanied by zero or more files

OVF descriptor OVF XML descriptor file

platform see deployment platform

virtual appliance a service delivered as a complete software stack installed on one or more virtual machines.

A virtual appliance is typically expected to be delivered in an OVF package.

virtual hardware the hardware (including the CPU, controllers, Ethernet devices, and disks) that is seen by the guest software.
3.20 virtual machine
the complete environment that supports the execution of guest software
A virtual machine is a full encapsulation of the virtual hardware, virtual disks, and the metadata associated with it. Virtual machines allow multiplexing of the underlying physical machine through a software layer called a hypervisor.

3.21 virtual machine collection
a service comprised of a set of virtual machines
The service can be a simple set of one or more virtual machines, or it can be a complex service built out of a combination of virtual machines and other virtual machine collections. Because virtual machine collections can be composed, it enables complex nested components.

4 Symbols and Abbreviated Terms
The following symbols and abbreviations are used in this document.

4.1.1 CIM
Common Information Model

4.1.2 IP
Internet Protocol

4.1.3 OVF
Open Virtualization Format

4.1.4 VM
Virtual Machine

5 OVF Packages

5.1 OVF Package Structure
An OVF package shall consist of the following files:

- one OVF descriptor with extension .ovf
- zero or one OVF manifest with extension .mf
- zero or one OVF certificate with extension .cert
- zero or more disk image files
- zero or more additional resource files, such as ISO images

The file extensions .ovf, .mf and .cert shall be used.

EXAMPLE 1: The following list of files is an example of an OVF package:

package.ovf
An OVF package can be stored as either a single unit or a set of files, as described in 5.3 and 5.4. Both modes shall be supported.

An OVF package may have a manifest file containing the SHA digests of individual files in the package. OVF packages authored according to this version of the specification shall use SHA256 digests; older OVF packages are allowed to use SHA1. The manifest file shall have an extension .mf and the same base name as the .ovf file and be a sibling of the .ovf file. If the manifest file is present, a consumer of the OVF package shall verify the digests by computing the actual SHA digests and comparing them with the digests listed in the manifest file. The manifest file shall contain SHA digests for all distinct files referenced in the References element of the OVF descriptor, see clause 7.1, and for no other files.

The syntax definitions below use ABNF with the exceptions listed in ANNEX A.

The format of the manifest file is as follows:

```
manifest_file = *( file_digest )
file_digest  = algorithm "(" file_name ")" "=" sp digest nl
algorithm    = "SHA1" | "SHA256"
digest       = *( hex-digit )
hex-digit    = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" | "a" | "b" | "c" | "d" | "e" | "f"
sp           = %x20
nl           = %x0A
```

**EXAMPLE 2:** The following example show the partial contents of a manifest file:

```
SHA256(package.ovf)= 9902cc5ec4f4a00cabbbf7b6d0d39263587ab430d5fbd65cd5e8707391c904
SHA256(vmdisk.vmdk)= aab66c4d70e17c6c2236a651a3fc618cafc5ec6424122904dc0b9c286fce40c
```

An OVF package may be signed by signing the manifest file. The digest of the manifest file is stored in a certificate file with extension .cert located along with the base64-encoded X.509 certificate. The .cert file shall have the same base name as the .ovf file and be a sibling of the .ovf file. A consumer of the OVF package shall verify the signature and should validate the certificate. The format of the certificate file shall be as follows:

```
certificate_file   = manifest_digest certificate_part
manifest_digest   = algorithm "(" file_name ")" "=" sp signed_digest nl
algorithm         = "SHA1" | "SHA256"
signed_digest     = *( hex-digit )
certificate_part  = certificate_header certificate_body certificate_footer
certificate_header = "-----BEGIN CERTIFICATE-----" nl
certificate_footer = "-----END CERTIFICATE-----" nl
certificate_body  = base64-encoded-certificate nl
; base64-encoded-certificate is a base64-encoded X.509 certificate, which may be split across multiple lines
hex-digit         = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" | "a" | "b" | "c" | "d" | "e" | "f"
sp                 = %x20
nl                 = %x0A
```

---

Work in Progress Specification - not a DMTF Standard

Version 2.0.0c
EXAMPLE 3:  The following list of files is an example of a signed OVF package:

```
package.ovf
package.mf
package.cert
de-DE-resources.xml
vmdisk1.vmdk
vmdisk2.vmdk
resource.iso
```

EXAMPLE 4:  The following example shows the contents of a sample OVF certification file, where the SHA1 digest of the manifest file has been signed with a 512 bit key:

```
SHA1(package.mf)= 7f4b8efb8fe20c6df1d68281a63f1b088e19dbf00e5af9db5e8e3e319de

-----BEGIN CERTIFICATE-----
MIIBgjCCASwCAQQwDQYJKoZIhvcNAQEEBQAwODELMAkGA1UEBhMCQVUxDDAKBgNV
BAgTA1FMRDEbMBkGA1UEAxMSU1NMZWF5L3JzYSB0ZXN0IENBMB4XDTk1MTAwOTIz
MzI1MzA1MB4XDTk4MDcwNTIzMzIwNVowYDelMAkGA1UECwYXVzdG9tMCcGA1UdDAs
EgQIgAUNDAMBgNVHAwTC1NMZWF5L3JzYWxlLm9uZDEwMB0GA1UEBxQXVzdG9tMIIB
BhMCQXZjBhZ2V0IENBMB4XDTk1MTAwOTIzMzI1MzA1MB4XDTk4MDcwNTIzMzIwNVow
YDelMAkGA1UECwYXVzdG9tMCcGA1UdDAsEgQIgAUNDAMBgNVHAwTC1NMZWF5L3JzYW
xlLm9uZDEwMB0GA1UEBxQXVzdG9tMIIBBhMCQXZjBhZ2V0IENBMB4XDTk1MTAwOTIz
MzI1MzA1MB4XDTk4MDcwNTIzMzIwNVowYDelMAkGA1UECwYXVzdG9tMCcGA1UdDAs
EgQIgAUNDAMBgNVHAwTC1NMZWF5L3JzYWxlLm9uZDEwMB0GA1UEBxQXVzdG9tMIIB
BhMCQXZjBhZ2V0IENBMB4XDTk1MTAwOTIzMzI1MzA1MB4XDTk4MDcwNTIzMzIwNVow
YDelMAkGA1UECwYXVzdG9tMCcGA1UdDAsEgQIgAUNDAMBgNVHAwTC1NMZWF5L3JzYW
xlLm9uZDEwMB0GA1UEBxQXVzdG9tMIIBbGkCAQRGIEQIgAUNDAMBgNVHAwTC1NMZWF5
L3JzYWxlLm9uZDEwMB0GA1UEBxQXVzdG9tMIIBbGkCAQRGIEQIgAUNDAMBgNVHAwTC1
NMZWF5L3JzYWxlLm9uZDEwMB0GA1UEBxQXVzdG9tMIIBbGkCAQRGIEQIgAUNDAMBgNV
HAwTC1NMZWF5L3JzYWxlLm9uZDEwMB0GA1UEBxQXVzdG9tMIIBbGkCAQRGIEQIgAUND
-----END CERTIFICATE-----
```

The manifest and certificate files, when present, shall not be included in the References section of the OVF descriptor (see 7.1). This ensures that the OVF descriptor content does not depend on whether the OVF package has a manifest or is signed, and the decision to add a manifest or certificate to a package can be deferred to a later stage.

The file extensions .mf and .cert may be used for other files in an OVF package, as long as they do not occupy the sibling URLs or path names where they would be interpreted as the package manifest or certificate.

### 5.2 Virtual Disk Formats

OVF does not require any specific disk format to be used, but to comply with this specification the disk format shall be given by a URI which identifies an unencumbered specification on how to interpret the disk format. The specification need not be machine readable, but it shall be static and unique so that the URI may be used as a key by software reading an OVF package to uniquely determine the format of the disk. The specification shall provide sufficient information so that a skilled person can properly interpret the disk format for both reading and writing of disk data. It is recommended that these URIs are resolvable.

### 5.3 Distribution as a Single File

An OVF package may be stored as a single file using the TAR format. The extension of that file shall be .ova (open virtual appliance or application).

EXAMPLE: The following example shows a sample filename for an OVF package of this type:

```
D:\virtualappliances\myapp.ova
```
For OVF packages stored as single file, all file references in the OVF descriptor shall be relative-path references and shall point to files included in the TAR archive. Relative directories inside the archive are allowed, but relative-path references shall not contain "." dot-segments.

Ordinarily, a TAR extraction tool would have to scan the whole archive, even if the file requested is found at the beginning, because replacement files can be appended without modifying the rest of the archive. For OVF TAR files, duplication is not allowed within the archive. In addition, the files shall be in the following order inside the archive:

1) OVF descriptor
2) OVF manifest (optional)
3) OVF certificate (optional)
4) The remaining files shall be in the same order as listed in the References section (see 7.1).
   Note that any external string resource bundle files for internationalization shall be first in the References section (see clause 10).
5) OVF manifest (optional)
6) OVF certificate (optional)

Note that the certificate file is optional. If no certificate file is present, the manifest file is also optional. If the manifest or certificate files are present, they shall either both be placed after the OVF descriptor, or both be placed at the end of the archive. If both manifest and certificate files are present, then the certificate file shall be immediately after the manifest file.

For deployment, the ordering restriction ensures that it is possible to extract the OVF descriptor from an OVF TAR file without scanning the entire archive. For generation, the ordering restriction ensures that an OVF TAR file can easily be generated on-the-fly. The restrictions do not prevent OVF TAR files from being created using standard TAR packaging tools.

The TAR format used shall comply with the USTAR (Uniform Standard Tape Archive) format as defined by the ISO/IEC/IEEE 9945:2009.

5.4 Distribution as a Set of Files

An OVF package can be made available as a set of files, for example on a standard Web server.

EXAMPLE: An example of an OVF package as a set of files on Web server follows:

   http://mywebsite/virtualappliances/package.ovf
   http://mywebsite/virtualappliances/vmdisk1.vmdk
   http://mywebsite/virtualappliances/vmdisk2.vmdk
   http://mywebsite/virtualappliances/resource.iso
   http://mywebsite/virtualappliances/de-DE-resources.xml

6 OVF Descriptor

All metadata about the package and its contents is stored in the OVF descriptor. This is an extensible XML document for encoding information, such as product details, virtual hardware requirements, and licensing.

The DMTF DSP8023 schema definition file for the OVF descriptor contains the elements and attributes. The OVF descriptor shall validate with the DMTF DSP8023 2.0.0 XML schema.

Clauses 7, 8, and 9, describe the semantics, structure, and extensibility framework of the OVF descriptor. These clauses are not a replacement for reading the schema definitions, but they complement the schema definitions.
The XML document of an OVF descriptor shall contain one Envelope element, which is the only element allowed at the top level.

The XML namespaces used in this specification are listed in Table 1. The choice of any namespace prefix is arbitrary and not semantically significant.

Table 1 – XML Namespace Prefixes

<table>
<thead>
<tr>
<th>Prefix</th>
<th>XML Namespace</th>
</tr>
</thead>
<tbody>
<tr>
<td>ovf</td>
<td><a href="http://schemas.dmtf.org/ovf/envelope/2">http://schemas.dmtf.org/ovf/envelope/2</a></td>
</tr>
<tr>
<td>ovfenv</td>
<td><a href="http://schemas.dmtf.org/ovf/environment/1">http://schemas.dmtf.org/ovf/environment/1</a></td>
</tr>
<tr>
<td>rasd</td>
<td><a href="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData">http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData</a></td>
</tr>
<tr>
<td>vssd</td>
<td><a href="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData">http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData</a></td>
</tr>
<tr>
<td>epasd</td>
<td><a href="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_EthernetPortAllocationSettingData">http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_EthernetPortAllocationSettingData</a></td>
</tr>
<tr>
<td>sasd</td>
<td><a href="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_StorageAllocationSettingData">http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_StorageAllocationSettingData</a></td>
</tr>
<tr>
<td>cim</td>
<td><a href="http://schemas.dmtf.org/wbem/wscim/1/common">http://schemas.dmtf.org/wbem/wscim/1/common</a></td>
</tr>
</tbody>
</table>

7 Envelope Element

The Envelope element describes all metadata for the virtual machines (including virtual hardware), as well as the structure of the OVF package itself.

The outermost level of the envelope consists of the following parts:

- A version indication, defined by the XML namespace URIs.
- A list of file references to all external files that are part of the OVF package, defined by the References element and its File child elements. These are typically virtual disk files, ISO images, and internationalization resources.
- A metadata part, defined by section elements, as defined in clause 9.
- A description of the content, either a single virtual machine (VirtualSystem element) or a collection of multiple virtual machines (VirtualSystemCollection element).
- A specification of message resource bundles for zero or more locales, defined by a Strings element for each locale.

EXAMPLE: An example of the structure of an OVF descriptor with the top-level Envelope element follows:

```xml
<?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/2"
          xmlns:"http://schemas.dmtf.org/ovf/envelope/2"
          xml:lang="en-US">
  <References>
```
The optional xml:lang attribute on the Envelope element shall specify the default locale for messages in the descriptor. The optional Strings elements shall contain string resource bundles for different locales. See clause 10 for more details on internationalization support.

7.1 File References

The file reference part defined by the References element allows a tool to easily determine the integrity of an OVF package without having to parse or interpret the entire structure of the descriptor. Tools can safely manipulate (for example, copy or archive) OVF packages with no risk of losing files.

External string resource bundle files for internationalization shall be placed first in the References element, see clause 10 for details.

Each File element in the reference part shall be given an identifier using the ovf:id attribute. The identifier shall be unique inside an OVF package. Each File element shall be specified using the ovf:href attribute, which shall contain a URL. Relative-path references and the URL schemes "file", "http", and "https" shall be supported, see RFC1738 and RFC3986. Other URL schemes should not be used. If no URL scheme is specified, the value of the ovf:href attribute shall be interpreted as a path name of the referenced file that is relative to the location of the OVF descriptor itself. The relative path name shall use the syntax of relative-path references in RFC3986. The referenced file shall exist. Two different File elements shall not reference the same file with their ovf:href attributes.
The size of the referenced file may be specified using the `ovf:size` attribute. The unit of this attribute is always bytes. If present, the value of the `ovf:size` attribute should match the actual size of the referenced file.

Each file referenced by a `File` element may be compressed using gzip (see RFC1952). When a `File` element is compressed using gzip, the `ovf:compression` attribute shall be set to "gzip". Otherwise, the `ovf:compression` attribute shall be set to "identity" or the entire attribute omitted. Alternatively, if the `href` is an HTTP or HTTPS URL, then the compression may be specified by the HTTP server by using the HTTP header Content-Encoding: gzip (see RFC2616). Using HTTP content encoding in combination with the `ovf:compression` attribute is allowed, but in general does not improve the compression ratio. When compression is used, the `ovf:size` attribute shall specify the size of the actual compressed file.

Files referenced from the reference part may be split into chunks to accommodate file size restrictions on certain file systems. Chunking shall be indicated by the presence of the `ovf:chunkSize` attribute; the value of `ovf:chunkSize` shall be the size of each chunk, except the last chunk, which may be smaller.

When `ovf:chunkSize` is specified, the `File` element shall reference a chunk file representing a chunk of the entire file. In this case, the value of the `ovf:href` attribute specifies only a part of the URL, and the syntax for the URL resolving to the chunk file is as follows. The syntax uses ABNF with the exceptions listed in ANNEX A.

```
chunk-url = href-value "." chunk-number
chunk-number = 9(decimal-digit)
decimal-digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
```

In this syntax, href-value is the value of the `ovf:href` attribute, and chunk-number is the 0-based position of the chunk starting with the value 0 and increases with increments of 1 for each chunk. Chunking can be combined with compression, the entire file shall be compressed before chunking and each chunk shall be an equal slice of the compressed file, except for the last chunk which may be smaller.

If the OVF package has a manifest file, the file name in the manifest entries shall match the value of the `ovf:href` attribute for the file, except if the file is split into multiple chunks, in which case the `chunk-url` shall be used, and the manifest file shall contain an entry for each individual chunk. For chunked files, the manifest file is allowed to contain an entry for the entire file; if present this digest shall also be verified.

**EXAMPLE 1:** The following example shows different types of file references:

```xml
<File ovf:id="disk1" ovf:href="disk1.vmdk"/>
<File ovf:id="disk2" ovf:href="disk2.vmdk" ovf:size="5368709120" ovf:chunkSize="2147483648"/>
<File ovf:id="iso1" ovf:href="resources/image1.iso"/>
<File ovf:id="iso2" ovf:href="http://mywebsite/resources/image2.iso"/>
```

**EXAMPLE 2:** The following example shows manifest entries corresponding to the file references above:

```xml
SHA1(disk1.vmdk) = 3e19644ec2e806f38951789c76f43e4a0ec7e233
SHA1(disk2.vmdk) = 4f715b73ff48b40f4217da248d47a2478e79d8
SHA1(disk2.vmdk) = 12849aeaf43e7a9550384d2b6b437bb8defaf
SHA1(disk2.vmdk) = 4cdd21424bd9eeaf4c42112876217de2ee5556d
SHA1(resources/image1.iso) = 72b37ff3fdd09f2a93f1b8395654649b6d06b5b3
SHA1(http://mywebsite/resources/image2.iso) =
d3c2d179011c970615c5cf10b30957d1c4c968ad
```
7.2 Content Element

Virtual machine configurations in an OVF package are represented by a VirtualSystem or VirtualSystemCollection element. These elements shall be given an identifier using the ovf:id attribute. Direct child elements of a VirtualSystemCollection shall have unique identifiers.

In the OVF schema, the VirtualSystem and VirtualSystemCollection elements are part of a substitution group with the Content element as head of the substitution group. The Content element is abstract and cannot be used directly. The OVF descriptor shall have one or more Content elements.

The VirtualSystem element describes a single virtual machine and is simply a container of section elements. These section elements describe virtual hardware, resources, and product information and are described in detail in clauses 8 and 9.

The structure of a VirtualSystem element is as follows:

```
<VirtualSystem ovf:id="simple-app">
  <Info>A virtual machine</Info>
  <Name>Simple Appliance</Name>
  <SomeSection>
    <!-- Additional section content -->
  </SomeSection>
  <!-- Additional sections can follow -->
</VirtualSystem>
```

The VirtualSystemCollection element is a container of multiple VirtualSystem or VirtualSystemCollection elements. Thus, arbitrary complex configurations can be described. The section elements at the VirtualSystemCollection level describe appliance information, properties, resource requirements, and so on, and are described in detail in clause 9.

The structure of a VirtualSystemCollection element is as follows:

```
<VirtualSystemCollection ovf:id="multi-tier-app">
  <Info>A collection of virtual machines</Info>
  <Name>Multi-tiered Appliance</Name>
  <SomeSection>
    <!-- Additional section content -->
  </SomeSection>
  <!-- Additional sections can follow -->
  <VirtualSystem ovf:id="...">
    <!-- Additional sections -->
  </VirtualSystem>
  <!-- Additional VirtualSystem or VirtualSystemCollection elements can follow-->
</VirtualSystemCollection>
```

All elements in the Content substitution group shall contain an Info element and may contain a Name element. The Info element contains a human readable description of the meaning of this entity. The Name element is an optional localizable display name of the content. See clause 10 for details on how to localize the Info and Name element.

7.3 Extensibility

This specification allows custom meta-data to be added to OVF descriptors in several ways:
• New section elements may be defined as part of the Section substitution group, and used where the OVF schemas allow sections to be present. All subtypes of Section contain an Info element that contains a human readable description of the meaning of this entity. The values of Info elements can be used, for example, to give meaningful warnings to users when a section is being skipped, even if the parser does not know anything about the section. See clause 10 for details on how to localize the Info element.

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

• The OVF schemas allow additional attributes on existing types.

Custom extensions shall not use XML namespaces defined in this specification. This applies to both custom elements and custom attributes.

On custom elements, a Boolean ovf:required attribute specifies whether the information in the element is required for correct behavior or optional. If not specified, the ovf:required attribute defaults to TRUE. A consumer of an OVF package that detects an extension that is required and that it does not understand shall fail.

For known Section elements, if additional child elements that are not understood are found and the value of their ovf:required attribute is TRUE, the consumer of the OVF package shall interpret the entire section as one it does not understand. The check is not recursive; it applies only to the direct children of the Section element.

This behavior ensures that older parsers reject newer OVF specifications, unless explicitly instructed not to do so.

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

EXAMPLE 1:

```xml
<!-- 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:

```xml
<!-- 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>
```

```xml
<!-- AnnotationSection extended with Author element -->
<otherns:Author ovf:required="false">John Smith</otherns:Author>
</AnnotationSection>
```

EXAMPLE 3:

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

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

- OVF descriptor uses only sections and elements and attributes that are defined in this specification. Conformance Level: 1.
- OVF descriptor uses custom sections or elements or attributes that are not defined in this specification, and all such extensions are optional as defined in 7.3. Conformance Level: 2.
- OVF descriptor uses custom sections or elements that are not defined in this specification and at least one such extension is required as defined in 7.3. The definition of all required extensions shall be publicly available in an open and unencumbered XML Schema. The complete specification may be inclusive in the XML schema or available as a separate document. Conformance Level: 3.

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

The conformance level is not specified directly in the OVF descriptor but shall be determined by the above rules.

8 Virtual Hardware Description

8.1 VirtualHardwareSection

Each VirtualSystem element may contain one or more VirtualHardwareSection elements, each of which describes the virtual hardware required by the virtual system. The virtual hardware required by a virtual machine is specified in VirtualHardwareSection elements. This specification supports abstract or incomplete hardware descriptions in which only the major devices are described. The hypervisor is allowed to create additional virtual hardware controllers and devices, as long as the required devices listed in the descriptor are realized.

This virtual hardware description is based on the CIM classes CIM_VirtualSystemSettingData, CIM_ResourceAllocationSettingData, CIM_EthernetPortAllocationSettingData, and CIM_StorageAllocationSettingData. The XML representation of the CIM model is based on the WS-CIM mapping (DSP0230). Note: This means that the XML elements that belong to the class complex type should be ordered by Unicode code point (binary) order of their CIM property name identifiers.

EXAMPLE: Example of VirtualHardwareSection:

```
<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>
  </Item>
</VirtualHardwareSection>
```
<rasd:Reservation>1</rasd:Reservation>
<rasd:ResourceType>3</rasd:ResourceType>
<rasd:VirtualQuantity>1</rasd:VirtualQuantity>
</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:ResourceType>4</rasd:ResourceType>
  <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
</Item>

<EthernetPortItem>
  <epasd:Address>00-16-8B-DB-00-5E</epasd:Address>
  <epasd:Connection>VM Network</epasd:Connection>
  <epasd:Description>Virtual NIC</epasd:Description>
  <epasd:ElementName>Ethernet Port</epasd:ElementName>
  <epasd:InstanceID>3</epasd:InstanceID>
  <epasd:NetworkPortProfileID>1</epasd:NetworkPortProfileID>
  <epasd:NetworkPortProfileIDType>4</epasd:NetworkPortProfileIDType>
  <epasd:ResourceType>10</epasd:ResourceType>
  <epasd:VirtualQuantityUnits>1</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:InstanceID>4</sasd:InstanceID>
  <sasd:Reservation>100</sasd:Reservation>
  <sasd:ResourceType>31</sasd:ResourceType>
  <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
</StorageItem>

A VirtualSystem element shall have a VirtualHardwareSection direct child element. VirtualHardwareSection is disallowed as a direct child element of a VirtualSystemCollection element and of an Envelope element.

Multiple VirtualHardwareSection element occurrences are allowed within a single VirtualSystem element. The consumer of the OVF package should select the most appropriate virtual hardware description for the particular virtualization platform. A VirtualHardwareSection element may contain...
an ovf:id attribute which can be used to identify the element. If present the attribute value must be unique within the VirtualSystem.

The ovf:transport attribute specifies the types of transport mechanisms by which properties are passed to the virtual machine in an OVF environment document. This attribute supports a pluggable and extensible architecture for providing guest/platform communication mechanisms. Several transport types may be specified separated by single space character. See 9.5 for a description of properties and clause 11 for a description of transport types and OVF environments.

A VirtualHardwareSection element contains sub elements that describe virtual system and virtual hardware resources (CPU, memory, network, and storage).

A VirtualHardwareSection element shall have zero or one System direct child element, followed by zero or more Item direct child elements, zero or more EthernetPortItem direct child elements, and zero or more StorageItem direct child elements.

The System element is an XML representation of the values of one or more properties of the CIM class CIM_VirtualSystemSettingData. The vssd:VirtualSystemType, a direct child element of System element, specifies a virtual system type identifier, which is an implementation defined string that uniquely identifies the type of the virtual system. For example, a virtual system type identifier could be vmx-4 for VMware's fourth-generation virtual hardware or xen-3 for Xen's third-generation virtual hardware. Zero or more virtual system type identifiers may be specified separated by single space character. In order for the OVF virtual system to be deployable on a target platform, the virtual machine on the target platform should support at least one of the virtual system types identified in the vssd:VirtualSystemType elements. The virtual system type identifiers specified in vssd:VirtualSystemType elements are expected to be matched against the values of property VirtualSystemTypesSupported of CIM class CIM_VirtualSystemManagementCapabilities.

The virtual hardware characteristics are described as a sequence of Item elements. The Item element is an XML representation of an instance of the CIM class CIM_ResourceAllocationSettingData. The element can describe all memory and CPU requirements as well as virtual hardware devices.

Multiple device subtypes may be specified in an Item element, separated by a single space character.

EXAMPLE:

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

The network hardware characteristics are described as a sequence of EthernetPortItem elements. The EthernetPortItem element is an XML representation of the values of one or more properties of the CIM class CIM_EthernetPortAllocationSettingData.

The storage hardware characteristics are described as a sequence of StorageItem elements. The StorageItem element is an XML representation of the values of one or more properties of the CIM class CIM_StorageAllocationSettingData.

### 8.2 Extensibility

The optional ovf:required attribute on the Item, EthernetPortItem, or StorageItem element specifies whether the realization of the element (for example, a CD-ROM or USB controller) is required for correct behavior of the guest software. If not specified, ovf:required defaults to TRUE.

On child elements of the Item, EthernetPortItem, or StorageItem element, the optional Boolean attribute ovf:required shall be interpreted, even though these elements are in a different RASD WS-CIM namespace. A tool parsing an Item element should act according to Table 2.
## Table 2 – Actions for Child Elements with `ovf:required` Attribute

<table>
<thead>
<tr>
<th>Child Element</th>
<th><code>ovf:required</code> Attribute Value</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Known</td>
<td>TRUE or not specified</td>
<td>Shall interpret Item, EthernetPortItem, or StorageItem</td>
</tr>
<tr>
<td>Known</td>
<td>FALSE</td>
<td>Shall interpret Item, EthernetPortItem, or StorageItem</td>
</tr>
<tr>
<td>Unknown</td>
<td>TRUE or not specified</td>
<td>Shall fail Item, EthernetPortItem, or StorageItem</td>
</tr>
<tr>
<td>Unknown</td>
<td>FALSE</td>
<td>Shall ignore Child Element</td>
</tr>
</tbody>
</table>

### 8.3 Virtual Hardware Elements


The child elements of `Item` represent the values of one or more properties exposed by the `CIM_ResourceAllocationSettingData` class. They have the semantics of defined settings as defined in [DSP1041](http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData.xsd), any profiles derived from [DSP1041](http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData.xsd) for specific resource types, and this document.

**EXAMPLE:** The following example shows a description of memory size:

```xml
<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>
```

The element type of the `EthernetPortItem` element in a `VirtualHardwareSection` element is `CIM_EthernetPortAllocationSettingData_Type` as defined in [http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_EthernetPortAllocationSettingData.xsd](http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_EthernetPortAllocationSettingData.xsd).

The child elements represent the values of one or more properties exposed by the `CIM_EthernetPortAllocationSettingData` class. They have the semantics of defined settings as defined in [DSP1050](http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_EthernetPortAllocationSettingData.xsd), any profiles derived from [DSP1050](http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_EthernetPortAllocationSettingData.xsd) for specific Ethernet port resource types, and this document.

**EXAMPLE:** The following example shows a description of a virtual Ethernet adapter:

```xml
<EthernetPortItem>
  <epasd:Address>00-16-8B-DB-00-5E</epasd:Address>
  <epasd:Connection>VM Network</epasd:Connection>
  <epasd:Description>Virtual NIC</epasd:Description>
  <epasd:ElementName>Ethernet Port 1</epasd:ElementName>
  <epasd:InstanceID>3</epasd:InstanceID>
  <epasd:NetworkPortProfileID>1</epasd:NetworkPortProfileID>
</EthernetPortItem>
```
The element type of the `StorageItem` element in a `VirtualHardwareSection` element is `CIM_StorageAllocationSettingData_Type` as defined in [http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_StorageAllocationSettingData.xsd](http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_StorageAllocationSettingData.xsd).

The child elements represent the values of one or more properties exposed by the `CIM_StorageAllocationSettingData` class. They have the semantics of defined settings as defined in DSP10xx, any profiles derived from DSP10xx for specific storage resource types, and this document.

**EXAMPLE:** The following example shows a description of a virtual storage:

```xml
<StorageItem>
  <sasd:AllocationUnits>byte*2^30</sasd:AllocationUnits>
  <sasd:Description>Virtual Disk</sasd:Description>
  <sasd:ElementName>100 GByte Virtual Disk</sasd:ElementName>
  <sasd:InstanceID>4</sasd:InstanceID>
  <sasd:Reservation>100</sasd:Reservation>
  <sasd:ResourceType>31</sasd:ResourceType>
  <sasd:VirtualQuantity>1</sasd:VirtualQuantity>
</StorageItem>
```

The `Description` element is used to provide additional metadata about the Item, EthernetPortItem, or `StorageItem` element itself. This element enables a consumer of the OVF package to provide descriptive information about all items, including items that were unknown at the time the application was written.

The `Caption`, `Description`, and `ElementName` elements are localizable using the `ovf:msgid` attribute from the OVF envelope namespace. See clause 10 for more details on internationalization support.

The optional `ovf:configuration` attribute contains a list of configuration names. See 9.8 on deployment options for semantics of this attribute. The optional `ovf:bound` attribute is used to specify ranges; see 8.4.

Devices such as disks, CD-ROMs, and networks need a backing from the deployment platform. The requirements on a backing are either specified using the `HostResource` or the `Connection` element.

For an Ethernet adapter, a logical network name is specified in the `Connection` element. Ethernet adapters that refer to the same logical network name within an OVF package shall be deployed on the same network.

The `HostResource` element is used to refer to resources included in the OVF descriptor as well as logical devices on the deployment platform. Values for `HostResource` elements referring to resources included in the OVF descriptor are formatted as URIs as specified in Table 3.

### Table 3 – HostResource Element

<table>
<thead>
<tr>
<th>Content</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ovf:/file/&lt;id&gt;</code></td>
<td>A reference to a file in the OVF, as specified in the References section. <code>&lt;id&gt;</code> shall be the value of the <code>ovf:id</code> attribute of the <code>File</code> element being referenced.</td>
</tr>
<tr>
<td><code>ovf:/disk/&lt;id&gt;</code></td>
<td>A reference to a virtual disk, as specified in the DiskSection or SharedDiskSection.</td>
</tr>
</tbody>
</table>
873  If no backing is specified for a device that requires a backing, the deployment platform shall make an appropriate choice, for example, by prompting the user. Specifying more than one backing for a device is not allowed.

876  Table 4 gives a brief overview on how elements from rasd, epasd, and sasd namespaces are used to describe virtual devices and controllers.

878  Table 4 – Elements for Virtual Devices and Controllers

<table>
<thead>
<tr>
<th>Element</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>A human-readable description of the meaning of the information. For example, “Specifies the memory size of the virtual machine”.</td>
</tr>
<tr>
<td>ElementName</td>
<td>A human-readable description of the content. For example, “256MB memory”.</td>
</tr>
<tr>
<td>InstanceID</td>
<td>A unique instance ID of the element within the section.</td>
</tr>
<tr>
<td>HostResource</td>
<td>Abstractly specifies how a device shall connect to a resource on the deployment platform. Not all devices need a backing. See Table 3.</td>
</tr>
<tr>
<td>ResourceType</td>
<td>Specifies the kind of device that is being described.</td>
</tr>
<tr>
<td>OtherResourceType</td>
<td></td>
</tr>
<tr>
<td>ResourceSubtype</td>
<td></td>
</tr>
<tr>
<td>AutomaticAllocation</td>
<td>For devices that are connectable, such as floppies, CD-ROMs, and Ethernet adaptors, this element specifies whether the device should be connected at power on.</td>
</tr>
<tr>
<td>Parent</td>
<td>The InstanceID of the parent controller (if any).</td>
</tr>
<tr>
<td>Connection</td>
<td>For an Ethernet adapter, this specifies the abstract network connection name for the virtual machine. All Ethernet adapters that specify the same abstract network connection name within an OVF package shall be deployed on the same network. The abstract network connection name shall be listed in the NetworkSection at the outermost envelope level.</td>
</tr>
<tr>
<td>Address</td>
<td>Device specific. For an Ethernet adapter, this specifies the MAC address.</td>
</tr>
<tr>
<td>AddressOnParent</td>
<td>For a device, this specifies its location on the controller.</td>
</tr>
<tr>
<td>AllocationUnits</td>
<td>Specifies the unit of allocation used. For example, “byte * 2^20”.</td>
</tr>
<tr>
<td>VirtualQuantity</td>
<td>Specifies the quantity of resources presented. For example, “256”.</td>
</tr>
<tr>
<td>Reservation</td>
<td>Specifies the minimum quantity of resources guaranteed to be available.</td>
</tr>
<tr>
<td>Limit</td>
<td>Specifies the maximum quantity of resources that are granted.</td>
</tr>
<tr>
<td>Weight</td>
<td>Specifies a relative priority for this allocation in relation to other allocations.</td>
</tr>
</tbody>
</table>

879  Only fields directly related to describing devices are mentioned. Refer to the CIM MOF for a complete description of all fields, each field corresponds to the identically named property in the CIM_ResourceAllocationSettingData class or a class derived from it.

882  8.4 Ranges on Elements

883  The optional ovf:bound attribute may be used to specify ranges for the Item elements. A range has a minimum, normal, and maximum value, denoted by min, normal, and max, where min <= normal <= max. The default values for min and max are those specified for normal.
A platform deploying an OVF package is recommended to start with the normal value and adjust the value within the range for ongoing performance tuning and validation.

For the Item, EthernetPortItem, and StorageItem elements in VirtualHardwareSection and ResourceAllocationSection elements, the following additional semantics are defined:

- Each Item, EthernetPortItem, or StorageItem element has an optional ovf:bound attribute. This value may be specified as min, max, or normal. The value defaults to normal. If the attribute is not specified or is specified as normal, then the item is interpreted as being part of the regular virtual hardware or resource allocation description.

- If the ovf:bound value is specified as either min or max, the item is used to specify the upper or lower bound for one or more values for a given InstanceID. Such an item is called a range marker.

The semantics of range markers are as follows:

- InstanceID and ResourceType shall be specified, and the ResourceType shall match other Item elements with the same InstanceID.

- Specifying more than one min range marker or more than one max range marker for a given RASD, EPASD, or SASD (identified with InstanceID) is invalid.

- An Item, EthernetPortItem, or StorageItem element with a range marker shall have a corresponding Item, EthernetPortItem, or StorageItem element without a range marker, that is, an Item, EthernetPortItem, and StorageItem element with no ovf:bound attribute or ovf:bound attribute with value normal. This corresponding item specifies the default value.

- For an Item, EthernetPortItem, and StorageItem element where only a min range marker is specified, the max value is unbounded upwards within the set of valid values for the property.

- For an Item, EthernetPortItem, and StorageItem where only a max range marker is specified, the min value is unbounded downwards within the set of valid values for the property.

- The default value shall be inside the range.

- The use of non-integer elements in range marker RASD, EPASD, or SASD is invalid.

EXAMPLE: The following example shows the use of range markers:

```xml
<VirtualHardwareSection>
  <Info>...</Info>
  <Item>
    <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
    <rasd:ElementName>512 MB memory size</rasd:ElementName>
    <rasd:InstanceID>0</rasd:InstanceID>
    <rasd:ResourceType>4</rasd:ResourceType>
    <rasd:VirtualQuantity>512</rasd:VirtualQuantity>
  </Item>
  <Item ovf:bound="min">
    <rasd:AllocationUnits>byte * 2^20</rasd:AllocationUnits>
    <rasd:ElementName>384 MB minimum memory size</rasd:ElementName>
    <rasd:InstanceID>0</rasd:InstanceID>
    <rasd:Reservation>384</rasd:Reservation>
  </Item>
</VirtualHardwareSection>
```
Core Metadata Sections in version 2

Table 5 shows the core metadata sections that are defined in the ovf namespace.

Table 5 – Core Metadata Sections in version 2

<table>
<thead>
<tr>
<th>Section</th>
<th>Locations</th>
<th>Multiplicity</th>
</tr>
</thead>
<tbody>
<tr>
<td>DiskSection</td>
<td>Envelope</td>
<td>Zero or one</td>
</tr>
<tr>
<td>Describes meta-information about all virtual disks in the package</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NetworkSection</td>
<td>Envelope</td>
<td>Zero or one</td>
</tr>
<tr>
<td>Describes logical networks used in the package</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ResourceAllocationSection</td>
<td>VirtualSystemCollection</td>
<td>Zero or one</td>
</tr>
<tr>
<td>Specifies reservations, limits, and shares on a given resource, such as memory or CPU for a virtual machine collection</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AnnotationSection</td>
<td>VirtualSystem</td>
<td>Zero or one</td>
</tr>
<tr>
<td>Specifies a free-form annotation on an entity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ProductSection</td>
<td>VirtualSystem</td>
<td>Zero or more</td>
</tr>
<tr>
<td>Specifies product-information for a package, such as product name and version, along with a set of properties that can be configured</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EulaSection</td>
<td>VirtualSystem</td>
<td>Zero or more</td>
</tr>
<tr>
<td>Specifies a license agreement for the software in the package</td>
<td></td>
<td></td>
</tr>
<tr>
<td>StartupSection</td>
<td>VirtualSystemCollection</td>
<td>Zero or one</td>
</tr>
<tr>
<td>Specifies how a virtual machine collection is powered on</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeploymentOptionSection</td>
<td>Envelope</td>
<td>Zero or one</td>
</tr>
<tr>
<td>Specifies a discrete set of intended resource requirements</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OperatingSystemSection</td>
<td>VirtualSystem</td>
<td>Zero or one</td>
</tr>
<tr>
<td>Specifies the installed guest operating system of a virtual machine</td>
<td></td>
<td></td>
</tr>
<tr>
<td>InstallSection</td>
<td>VirtualSystem</td>
<td>Zero or one</td>
</tr>
<tr>
<td>Specifies that the virtual machine needs to be initially booted to install and configure the software</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EnvironmentFilesSection</td>
<td>VirtualSystem</td>
<td>Zero or one</td>
</tr>
<tr>
<td>Specifies additional files from an OVF package to be included in the OVF environment</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The following subclauses describe the semantics of the core sections and provide some examples. The sections are used in several places of an OVF envelope; the description of each section defines where it may be used. See the OVF schema for a detailed specification of all attributes and elements.

In the OVF schema, all sections are part of a substitution group with the Section element as head of the substitution group. The Section element is abstract and cannot be used directly.

### 9.1 DiskSection

A DiskSection describes meta-information about virtual disks in the OVF package. Virtual disks and their metadata are described outside the virtual hardware to facilitate sharing between virtual machines within an OVF package. Virtual disks in DiskSection can be referenced by multiple virtual machines, but seen from the guest software each virtual machine get individual private disks. Any level of sharing done at runtime is deployment platform specific and not visible to the guest software. See clause 9.13 for details on how to configure sharing of virtual disk at runtime with concurrent access.

**EXAMPLE:** The following example shows a description of virtual disks:

```xml
<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 ovf:diskId="vmdisk3" ovf:capacity="${disk.size}"
    ovf:capacityAllocationUnits="byte * 2^30">
  </Disk>
</DiskSection>
```

DiskSection is a valid section at the outermost envelope level only.
Each virtual disk is represented by a Disk element that shall be given an identifier using the ovf:diskId attribute; the identifier shall be unique within the DiskSection.

The capacity of a virtual disk shall be specified by the ovf:capacity attribute with an xs:long integer value. The default unit of allocation shall be bytes. The optional string attribute ovf:capacityAllocationUnits may be used to specify a particular unit of allocation. Values for ovf:capacityAllocationUnits shall match the format for programmatic units defined in DSP0004 with the restriction that the base unit shall be "byte".

The ovf:fileRef attribute denotes the virtual disk content by identifying an existing File element in the References element, the File element is identified by matching its ovf:id attribute value with the ovf:fileRef attribute value. Omitting the ovf:fileRef attribute shall indicate an empty disk. In this case, the disk shall be created and the entire disk content zeroed at installation time. The guest software will typically format empty disks in some file system format.

The format URI (see 5.2) of a non-empty virtual disk shall be specified by the ovf:format attribute.

Different Disk elements shall not contain ovf:fileRef attributes with identical values. Disk elements shall be ordered such that they identify any File elements in the same order as these are defined in the References element.

For empty disks, rather than specifying a fixed virtual disk capacity, the capacity for an empty disk may be given using an OVF property, for example ovf:capacity="${disk.size}". The OVF property shall resolve to an xs:long integer value. See 9.5 for a description of OVF properties. The ovf:capacityAllocationUnits attribute is useful when using OVF properties because a user may be prompted and can then enter disk sizing information in, for example, gigabytes.

For non-empty disks, the actual used size of the disk may optionally be specified using the ovf:populatedSize attribute. The unit of this attribute is always bytes. ovf:populatedSize is allowed to be an estimate of used disk size but shall not be larger than ovf:capacity.

In VirtualHardwareSection, virtual disk devices may have a rasd:HostResource element referring to a Disk element in DiskSection; see 8.3. The virtual disk capacity shall be defined by the ovf:capacity attribute on the Disk element. If a rasd:VirtualQuantity element is specified along with the rasd:HostResource element, the virtual quantity value shall not be considered and may have any value.

OVF allows a disk image to be represented as a set of modified blocks in comparison to a parent image. The use of parent disks can often significantly reduce the size of an OVF package if it contains multiple disks with similar content, such as a common base operating system. Actual sharing of disk blocks at runtime is optional and deployment platform specific and shall not be visible to the guest software.

For the Disk element, a parent disk may optionally be specified using the ovf:parentRef attribute, which shall contain a valid ovf:diskId reference to a different Disk element. If a disk block does not exist locally, lookup for that disk block then occurs in the parent disk. In DiskSection, parent Disk elements shall occur before child Disk elements that refer to them. Similarly, in References element, the File elements referred from these Disk elements shall respect the same ordering. The ordering restriction ensures that in an OVA archive, parent disks always occur before child disks, making it possible for a tool to consume the archive in a streaming mode, see also clause 5.3.

9.2 NetworkSection

The NetworkSection element shall list all logical networks used in the OVF package.

```xml
<NetworkSection>
  <Info>List of logical networks used in the package</Info>
</NetworkSection>
```
NetworkSection is a valid element at the outermost envelope level.

All networks referred to from Connection elements in all VirtualHardwareSection elements shall be defined in the NetworkSection.

Starting with version 2.0 of this specification, each logical network may contain a set of networking attributes that should be applied when mapping the logical network at deployment time to a physical or virtual network. Networking attributes are specified by embedding or referencing zero or more instances of network port profile as specified by NetworkPortProfile or NetworkPortProfileRef child element of the Network element.

The NetworkPortProfile child element of the Network element defines the contents of a network port profile. The NetworkPortProfileRef child element of the Network element defines the reference to a network port profile.

### 9.3 ResourceAllocationSection

The ResourceAllocationSection element describes all resource allocation requirements of a VirtualSystemCollection entity. These resource allocations shall be performed when deploying the OVF package.

```xml
<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:configuration="..." ovf:bound="...">
        <rasd:AllocationUnits>hertz * 10^6</rasd:AllocationUnits>
        <rasd:ElementName>500 MHz reservation</rasd:ElementName>
    </Item>
</ResourceAllocationSection>
```
ResourceAllocationSection is a valid element for a VirtualSystemCollection entity.

The optional ovf:configuration attribute contains a list of configuration names. See 9.8 on deployment options for semantics of this attribute.

The optional ovf:bound attribute contains a value of min, max, or normal. See 8.4 for semantics of this attribute.

9.4 AnnotationSection

The AnnotationSection element is a user-defined annotation on an entity. Such annotations may be displayed when deploying the OVF package.

AnnotationSection is a valid element for a VirtualSystem and a VirtualSystemCollection entity.

See clause 10 for details on how to localize the Annotation element.
9.5 ProductSection

The ProductSection element specifies product-information for an appliance, such as product name, version, and vendor.

```xml
<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="admin.email" ovf:type="string" ovf:userConfigurable="true">
      <Label>Admin email</Label>
      <Description>Email address of administrator</Description>
    </Property>
    <Category>Admin properties</Category>
    <Property ovf:key="app_log" ovf:type="string" ovf:value="low" ovf:userConfigurable="true">
      <Description>Loglevel for the service</Description>
    </Property>
    <Property ovf:key="app_isSecondary" ovf:value="false" ovf:type="boolean">
      <Description>Cluster setup for application server</Description>
    </Property>
    <Property ovf:key="app_ip" ovf:type="string" ovf:value="${appserver-vm}">
      <Description>IP address of the application server VM</Description>
    </Property>
  </Icon>
</ProductSection>
```

The optional Product element specifies the name of the product, while the optional Vendor element specifies the name of the product vendor. The optional Version element specifies the product version in short form, while the optional FullVersion element describes the product version in long form. The optional ProductUrl element specifies a URL which shall resolve to a human readable description of the product, while the optional VendorUrl specifies a URL which shall resolve to a human readable description of the vendor.

The optional AppUrl element specifies a URL resolving to the deployed product instance; this element is experimental. The optional Icon element specifies display icons for the product; this element is experimental.

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

ProductSection is a valid section for a VirtualSystem and a VirtualSystemCollection entity.

Property elements may be grouped by using Category elements. The set of Property elements grouped by a Category element is the sequence of Property elements following the Category element, until but not including an element that is not a Property element. For OVF packages containing a large number of Property elements, this may provide a simpler installation experience.
Similarly, each Property element may have a short label defined by its Label child element in addition to a description defined by its Description child element. See clause 10 for details on how to localize the Category element and the Description and Label child elements of the Property element.

Each Property element in a ProductSection shall be given an identifier that is unique within the ProductSection using the ovf:key attribute.

Each Property element in a ProductSection shall be given a type using the ovf:type attribute and optionally type qualifiers using the ovf:qualifiers attribute. Valid types are listed in Table 6, and valid qualifiers are listed in Table 7.

The optional attribute ovf:value is used to provide a default value for a property. One or more optional Value elements may be used to define alternative default values for specific configurations, as defined in 9.8.

The optional attribute ovf:userConfigurable determines whether the property value is configurable during the installation phase. If ovf:userConfigurable is FALSE or omitted, the ovf:value attribute specifies the value to be used for that customization parameter during installation. If ovf:userConfigurable is TRUE, the ovf:value attribute specifies a default value for that customization parameter, which may be changed during installation.

A simple OVF implementation such as a command-line installer typically uses default values for properties and does not prompt even though ovf:userConfigurable is set to TRUE. To force prompting at startup time, omitting the ovf:value attribute is sufficient for integer types, because the empty string is not a valid integer value. For string types, prompting may be forced by adding a qualifier requiring a non-empty string, see Table 7.

The optional Boolean attribute ovf:password indicates that the property value may contain sensitive information. The default value is FALSE. OVF implementations prompting for property values are advised to obscure these values when ovf:password is set to TRUE. This is similar to HTML text input of type password. Note that this mechanism affords limited security protection only. Although sensitive values are masked from casual observers, default values in the OVF descriptor and assigned values in the OVF environment are still passed in clear text.

Zero or more ProductSections may be specified within a VirtualSystem or VirtualSystemCollection. Typically, a ProductSection corresponds to a particular software product that is installed. Each product section at the same entity level shall have a unique ovf:class and ovf:instance attribute pair. For the common case where only a single ProductSection is used, the ovf:class and ovf:instance attributes are optional and default to the empty string. It is recommended that the ovf:class property be used to uniquely identify the software product using the reverse domain name convention. Examples of values are com.vmware.tools and org.apache.tomcat. If multiple instances of the same product are installed, the ovf:instance attribute is used to identify the different instances.

Property elements are exposed to the guest software through the OVF environment, as described in clause 11. The value of the ovfenv: key attribute of a Property element exposed in the OVF environment shall be constructed from the value of the ovf: key attribute of the corresponding Property element defined in a ProductSection entity of an OVF descriptor as follows:

key-value-env = [class-value "."] key-value-prod ["." instance-value]

The syntax definition above use ABNF with the exceptions listed in ANNEX A, where:

- class-value is the value of the ovf:class attribute of the Property element defined in the ProductSection entity. The production [class-value "."] shall be present if and only if class-value is not the empty string.
key-value-prod is the value of the ovf:key attribute of the Property element defined in the ProductSection entity.

instance-value is the value of the ovf:instance attribute of the Property element defined in the ProductSection entity. The production ["."] instance-value] shall be present if and only if instance-value is not the empty string.

EXAMPLE: The following OVF environment example shows how properties can be propagated to the guest software:

```xml
<Property ovf:key="com.vmware.tools.logLevel" ovf:value="none"/>
<Property ovf:key="org.apache.tomcat.logLevel.1" ovf:value="debug"/>
<Property ovf:key="org.apache.tomcat.logLevel.2" ovf:value="normal"/>
```

The consumer of an OVF package should prompt for properties where ovf:userConfigurable is TRUE. These properties may be defined in multiple ProductSections as well as in sub-entities in the OVF package.

If a ProductSection exists, then the first ProductSection entity defined in the top-level Content element of a package shall define summary information that describes the entire package. After installation, a consumer of the OVF package could choose to make this information available as an instance of the CIM_Product class.

Property elements specified on a VirtualSystemCollection are also seen by its immediate children (see clause 11). Children may refer to the properties of a parent VirtualSystemCollection using macros on the form ${name} as value for ovf:value attributes.

Table 6 lists the valid types for properties. These are a subset of CIM intrinsic types defined in DSP0004, which also define the value space and format for each intrinsic type. Each Property element shall specify a type using the ovf:type attribute.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint8</td>
<td>Unsigned 8-bit integer</td>
</tr>
<tr>
<td>sint8</td>
<td>Signed 8-bit integer</td>
</tr>
<tr>
<td>uint16</td>
<td>Unsigned 16-bit integer</td>
</tr>
<tr>
<td>sint16</td>
<td>Signed 16-bit integer</td>
</tr>
<tr>
<td>uint32</td>
<td>Unsigned 32-bit integer</td>
</tr>
<tr>
<td>sint32</td>
<td>Signed 32-bit integer</td>
</tr>
<tr>
<td>uint64</td>
<td>Unsigned 64-bit integer</td>
</tr>
<tr>
<td>sint64</td>
<td>Signed 64-bit integer</td>
</tr>
<tr>
<td>String</td>
<td>String</td>
</tr>
<tr>
<td>Boolean</td>
<td>Boolean</td>
</tr>
<tr>
<td>real32</td>
<td>IEEE 4-byte floating point</td>
</tr>
<tr>
<td>real64</td>
<td>IEEE 8-byte floating point</td>
</tr>
</tbody>
</table>

Table 7 lists the supported CIM type qualifiers as defined in DSP0004. Each Property element may optionally specify type qualifiers using the ovf:qualifiers attribute with multiple qualifiers separated by commas; see production qualifierList in ANNEX A “MOF Syntax Grammar Description” in DSP0004.
Table 7 – Property Qualifiers

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>MinLen (min) MaxLen (max) ValueMap [...]</td>
</tr>
<tr>
<td>uint8</td>
<td>ValueMap [...]</td>
</tr>
<tr>
<td>sint8</td>
<td></td>
</tr>
<tr>
<td>uint16</td>
<td></td>
</tr>
<tr>
<td>sint16</td>
<td></td>
</tr>
<tr>
<td>uint32</td>
<td></td>
</tr>
<tr>
<td>sint32</td>
<td></td>
</tr>
<tr>
<td>uint64</td>
<td></td>
</tr>
<tr>
<td>sint64</td>
<td></td>
</tr>
</tbody>
</table>

9.6 EulaSection

A EulaSection contains the legal terms for using its parent Content element. This license shall be shown and accepted during deployment of an OVF package. Multiple EulaSections may be present in an OVF. If unattended installations are allowed, all embedded license sections are implicitly accepted.

```xml
<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. Suciis 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>
```

EulaSection is a valid section for a VirtualSystem and a VirtualSystemCollection entity.

See clause 10 for details on how to localize the License element.

See also clause 10 for description of storing EULA license contents in an external file without any XML header or footer. This allows inclusion of standard license or copyright text files in unaltered form.

9.7 StartupSection

The StartupSection specifies how a virtual machine collection is powered on and off.

```xml
<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>
```
Each Content element that is a direct child of a VirtualSystemCollection may have a corresponding Item element in the StartupSection entity of the VirtualSystemCollection entity. Note that Item elements may correspond to both VirtualSystem and VirtualSystemCollection entities. When a start or stop action is performed on a VirtualSystemCollection entity, the respective actions on the Item elements of its StartupSection entity are invoked in the specified order. Whenever an Item element corresponds to a (nested) VirtualSystemCollection entity, the actions on the Item elements of its StartupSection entity shall be invoked before the action on the Item element corresponding to that VirtualSystemCollection entity is invoked (i.e., depth-first traversal).

The following required attributes on Item are supported for a VirtualSystem and VirtualSystemCollection:

- ovf:id shall match the value of the ovf:id attribute of a Content element which is a direct child of this VirtualSystemCollection. That Content element describes the virtual machine or virtual machine collection to which the actions defined in the Item element apply.

- ovf:order specifies the startup order using non-negative integer values. The order of execution of the start action is the numerical ascending order of the values. Items with same order identifier may be started up concurrently. The order of execution of the stop action is the numerical descending order of the values.

The following optional attributes on Item are supported for a VirtualSystem:

- ovf:startDelay specifies a delay in seconds to wait until proceeding to the next order in the start sequence. The default value is 0.

- ovf:waitingForGuest enables the platform to resume the startup sequence after the guest software has reported it is ready. The interpretation of this is deployment platform specific. The default value is FALSE.

- ovf:startAction specifies the start action to use. Valid values are powerOn and none. The default value is powerOn.

- ovf:stopDelay specifies a delay in seconds to wait until proceeding to the previous order in the stop sequence. The default value is 0.

- ovf:stopAction specifies the stop action to use. Valid values are powerOff, guestShutdown, and none. The interpretation of guestShutdown is deployment platform specific. The default value is powerOff.

If not specified, an implicit default Item is created for each entity in the collection with ovf:order="0". Thus, for a trivial startup sequence no StartupSection needs to be specified.

9.8 DeploymentOptionSection

The DeploymentOptionSection specifies a discrete set of intended resource configurations. The author of an OVF package can include sizing metadata for different configurations. A consumer of the OVF shall select a configuration, for example, by prompting the user. The selected configuration is visible in the OVF environment, enabling guest software to adapt to the selected configuration. See clause 11.

The DeploymentOptionSection specifies an ID, label, and description for each configuration.
<DeploymentOptionSection>
  <Configuration ovf:id="minimal">
    <Label>Minimal</Label>
    <Description>Some description</Description>
  </Configuration>
  <Configuration ovf:id="normal" ovf:default="true">
    <Label>Typical</Label>
    <Description>Some description</Description>
  </Configuration>
</DeploymentOptionSection>

The DeploymentOptionSection has the following semantics:

- If present, the DeploymentOptionSection is valid only at the envelope level, and only one section shall be specified in an OVF descriptor.
- The discrete set of configurations is described with Configuration elements, which shall have identifiers specified by the ovf:id attribute that are unique in the package.
- A default Configuration element may be specified with the optional ovf:default attribute. If no default is specified, the first element in the list is the default. Specifying more than one element as the default is invalid.
- The Label and Description elements are localizable using the ovf:msgid attribute. See clause 10 for more details on internationalization support.

Configurations may be used to control resources for virtual hardware and for virtual machine collections. Item, EthernetPortItem, and StorageItem elements in VirtualHardwareSection elements describe resources for VirtualSystem entities, while Item, EthernetPortItem, and StorageItem elements in ResourceAllocationSection elements describe resources for virtual machine collections. For these two Item, EthernetPortItem, or StorageItem types, the following additional semantics are defined:

- Each Item EthernetPortItem, and StorageItem has an optional ovf:configuration attribute, containing a list of configurations separated by a single space character. If not specified, the item shall be selected for any configuration. If specified, the item shall be selected only if the chosen configuration ID is in the list. A configuration attribute shall not contain an ID that is not specified in the DeploymentOptionSection.
- Within a single VirtualHardwareSection or ResourceAllocationSection, multiple Item, EthernetPortItem, and StorageItem elements are allowed to refer to the same InstanceID. A single combined Item, EthernetPortItem, or StorageItem for the given InstanceID shall be constructed by picking up the child elements of each Item, EthernetPortItem, or StorageItem element, with child elements of a former Item, EthernetPortItem, or StorageItem element in the OVF descriptor not being picked up if there is a like-named child element in a latter Item, EthernetPortItem, or StorageItem element. Any attributes specified on child elements of Item, EthernetPortItem, or StorageItem elements that are not picked up that way, are not part of the combined Item, EthernetPortItem, or StorageItem element.
- All Item, EthernetPortItem, StorageItem elements shall specify ResourceType, and Item, EthernetPortItem, and StorageItem elements with the same InstanceID shall agree on ResourceType.
EXAMPLE 1: The following example shows a VirtualHardwareSection:

```
<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>
```

Note that the attributes `ovf:configuration` and `ovf:bound` on `Item` may be used in combination to provide very flexible configuration options.

Configurations can further be used to control default values for properties and whether properties are user configurable. For `Property` elements inside a `ProductSection`, the following additional semantic is defined:

- It is possible to specify alternative default property values for different configurations in a DeploymentOptionSection. In addition to a `Label` and `Description` element, each `Property` element may optionally contain `Value` elements. The `Value` element shall have an `ovf:value` attribute specifying the alternative default and an `ovf:configuration` attribute specifying the configuration in which this new default value should be used. Multiple `Value` elements shall not refer to the same configuration.

- Starting with version 2.0 of this specification, a `Property` element may optionally have an `ovf:configuration` attribute specifying the configuration in which this property should be user configurable. The value of `ovf:userConfigurable` is implicitly set to FALSE for all other configurations, in which case the default value of the property may not be changed during installation.

EXAMPLE 2: The following shows an example `ProductSection`:

```
<ProductSection>
  <Property ovf:key="app.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="app.log" ovf:type="string" ovf:value="low"
    ovf:userConfigurable="true">
  </Property>
</ProductSection>
```
In the example above, the app.adminEmail property is only user configurable in the standard configuration, while the default value for the app.log property is changed from low to none in the minimal configuration.

9.9 OperatingSystemSection

An OperatingSystemSection specifies the operating system installed on a virtual machine.

The valid values for ovf:id are defined by the ValueMap qualifier in the CIM_OperatingSystem.OsType property.

OperatingSystemSection is a valid section for a VirtualSystem entity only.

9.10 InstallSection

The InstallSection, if specified, indicates that the virtual machine needs to be booted once in order to install and/or configure the guest software. The guest software is expected to access the OVF environment during that boot, and to shut down after having completed the installation and/or configuration of the software, powering off the guest.

If the InstallSection is not specified, this indicates that the virtual machine does not need to be powered on to complete installation of guest software.

InstallSection is a valid section for a VirtualSystem entity only.

The optional ovf:initialBootStopDelay attribute specifies a delay in seconds to wait for the virtual machine to power off. If not set, the implementation shall wait for the virtual machine to power off by itself.

If the delay expires and the virtual machine has not powered off, the consumer of the OVF package shall indicate a failure.

Note that the guest software in the virtual machine can do multiple reboots before powering off.

Several VMs in a virtual machine collection may have an InstallSection defined, in which case the above step is done for each VM, potentially concurrently.

9.11 EnvironmentFilesSection

Clause 11 describes how the OVF environment file is used to deliver runtime customization parameters to the guest operating system. In version 1 of this specification, the OVF environment file is the only file delivered to the guest operating system outside of the virtual disk structure. In order to provide additional
deployment time customizations, EnvironmentFilesSection enable OVF package authors to specify additional files in the OVF package, outside of the virtual disks, that will also be provided to the guest operating system at runtime via a transport.

This enables increased flexibility in image customization outside of virtual disk capture, allowing OVF package authors to customize solutions by combining existing virtual disks without modifying them.

For each additional file provided to the guest, the EnvironmentFilesSection shall contain a File element with required attributes ovf:fileRef and ovf:path. The ovf:fileRef attribute shall denote the actual content by identifying an existing 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 denotes the relative location on the transport where this file will be placed, using the syntax of relative-path references in RFC3986.

The referenced File element in the References element identify the content using one of the URL schemes "file", "http", or "https". For the "file" scheme, the content is static and included in the OVF package. For the "http" and "https" schemes, the content shall be downloaded prior to the initial boot of the virtual system.

The iso transport shall support this mechanism, see clause 11.2 for details. For this transport, the root location relative to ovf:path values shall be directory ovffiles contained in the root directory of the ISO image. The guest software can access the information using standard guest operating system tools.

Other custom transport may support this mechanism. Custom transports will need to specify how to access multiple data sources from a root location.

EXAMPLE:

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

9.12 BootDeviceSection

Individual virtual machines will generally use the default device boot order provided by the virtualization platform's virtual BIOS. BootDeviceSection allows the OVF package author to specify particular boot configurations and boot order settings. This enables booting from non-default devices such as a NIC using PXE, a USB device or a secondary disk. Moreover there could be multiple boot configurations with
different boot orders. For example, a virtual disk may be need to be patched before it is bootable and a
patch ISO image could be included in the OVF package.

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 boot configuration is defined by the class
CIM_BootConfigSetting which in turn contains one or more CIM_BootSourceSetting classes as
defined in the WS-CIM schema. Each class representing the boot source in turn has either the specific
device or a “device type” such as disk or CD/DVD as a boot source.

In the context of this specification, the InstanceID field of CIM_BootSourceSetting is used for
identifying a specific device as the boot source. The InstanceID field of the device as specified in the
Item description of the device in the VirtualHardwareSection is used to specify the device as a
boot source. In case the source is desired to be a device type, the StructuredBootString field is
used to denote the type of device with values defined by the CIM boot control profile. When a boot source
is a device type, the deployment platform should try all the devices of the specified type.

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:

```xml
<Enveloped>
  ...
  <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>
      <bootc:CIM_BootSourceSetting>

        <bootc:Caption>Fix-up DVD on the network</bootc:Caption>
        <bootc:InstanceID>3</bootc:InstanceID> <!-- Network device-->

      </bootc:CIM_BootSourceSetting>

    </bootc:CIM_BootSourceSetting>

    <bootc:CIM_BootSourceSetting>

      <bootc:Caption>Boot virtual disk</bootc:Caption>
      <bootc:StructuredBootString>CIM:Hard-Disk</bootc:StructuredBootString>

    </bootc:CIM_BootSourceSetting>

    </bootc:CIM_BootConfigSetting>

  </ovf:BootDeviceSection>

  ...

  </VirtualSystem>

</Enveloped>
```

9.13 SharedDiskSection

The existing DiskSection in clause 9.1 describes virtual disks in the OVF package. Virtual disks in the
DiskSection can be referenced by multiple virtual machines, but seen from the guest software each
virtual machine gets individual private disks. Any level of sharing done at runtime is deployment platform
specific and not visible to the guest software.

Certain applications such as clustered databases rely on multiple virtual machines sharing the same
virtual disk at runtime. SharedDiskSection allows the OVF package author to specify Disk elements
shared by more than one VirtualSystem at runtime, these could be virtual disks backing by an external
File reference, or empty virtual disks without backing. It is recommended that the guest software use
cluster-aware file system technology to be able to handle concurrent access.

EXAMPLE:

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

SharedDiskSection is a valid section at the outermost envelope level only.

Each virtual disk is represented by a SharedDisk element that shall be given an identifier using the
ovf:diskId attribute; the identifier shall be unique within the combined content of DiskSection and
SharedDiskSection. The SharedDisk element has the same structure as the Disk element in
DiskSection, with the addition of an optional boolean attribute ovf:readOnly stating whether shared
disk access is read-write or read-only.

Shared virtual disks are referenced from virtual hardware using the using the HostResource element as
described in clause 8.3.

It is optional for the virtualization platform to support SharedDiskSection. The platform should give an
appropriate error message based on the value of the ovf:required attribute on the
SharedDiskSection element.

9.14 ScaleOutSection

The number of VirtualSystems and VirtualSystemCollections contained in an OVF package is generally
fixed and determined by the structure inside the Envelope element. The ScaleOutSection allows a
VirtualSystemCollection to contain a set of children that are homogeneous with respect to a prototypical
VirtualSystem or VirtualSystemCollection. The ScaleOutSection shall cause the deployment platform
to replicate the prototype a number of times, thus allowing the number of instantiated virtual machines to
be configured dynamically at deployment time.

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

EXAMPLE:
<Environment ...
  <ovfenv:id="web-server-1">
    ...
  </ovfenv:id="web-server-1">
This mechanism enables dynamic scaling of virtual machine instances at deployment time. Scaling at runtime is not within the scope of this specification.

ScaleOutSection is a valid section inside VirtualSystemCollection only.

The ovf:id attribute on ScaleOutSection identifies the VirtualSystem or VirtualSystemCollection prototype to be replicated.

For the InstanceCount element, the ovf:minimum and ovf:maximum attribute values shall be non-negative integers and ovf:minimum shall be less than or equal to the value of ovf:maximum. The ovf:minimum value may be zero in which case the VirtualSystemCollection may contain zero instances of the prototype. If the ovf:minimum attribute is not present, it is assumed to have a value of one. If the ovf:maximum attribute is not present, it is assumed to have a value of unbounded. The ovf:default attribute is required and shall contain a value within the range defined by ovf:minimum and ovf:maximum.

Each replicated instance shall be assigned a unique ovf:id value within the VirtualSystemCollection. The unique ovf:id value shall be constructed from the prototype ovf:id value with a sequence number appended as follows:

```
replica-ovf-id = prototype-ovf-id "-" decimal-number
```

The syntax definitions above use ABNF with the exceptions listed in ANNEX A. The first replica shall have sequence number one and following sequence numbers shall be incremented by one for each replica. Note that after deployment, no VirtualSystem will have the prototype ovf:id value itself.

If the prototype being replicated has a starting order in the StartupSection, all replicas shall share this value. It is not possible to specify a particular starting sequence among replicas.

Property values for Property elements in the prototype are prompted for once per replica created. If the OVF package author requires a property value to be shared among instances, that Property may be declared at the containing VirtualSystemCollection level and referenced by replicas as described in clause 9.5.

Configurations from the DeploymentOptionSection may be used to control values for InstanceCount. The InstanceCount element may have an ovf:configuration attribute specifying the configuration in which this element should be used. Multiple elements shall not refer to the same configuration, and a configuration attribute shall not contain an ovf:id value that is not specified in the DeploymentOptionSection.

EXAMPLE:

```
<VirtualSystemCollection ovf:id="web-tier">
  ...
  <DeploymentOptionSection>
    <Info>Deployment size options</Info>
    <Configuration ovf:id="minimal">
      <Label>Minimal</Label>
    </Configuration>
  </DeploymentOptionSection>
</VirtualSystemCollection>
```
1611  <Description>Minimal deployment scenario</Description>
1612  </Configuration>
1613  <Configuration ovf:id="common" ovf:default="true">
1614    <Label>Typical</Label>
1615    <Description>Common deployment scenario</Description>
1616  </Configuration>
1617  ...
1618  </DeploymentOptionSection>
1619  ...
1620  <ovf:ScaleOutSection ovf:id="web-server">
1621    <Info>Web tier</Info>
1622    <Description>Number of web server instances in web tier</Description>
1623    <ovf:InstanceCount ovf:default="4"/>
1624    <ovf:InstanceCount ovf:default="1" ovf:configuration="minimal"/>
1625  </ovf:ScaleOutSection>
1626  ...
1627  </VirtualSystemCollection>

In the example above, the default replica count is four, unless the minimal deployment scenario is chosen, in which case the default is one.

9.15 PlacementGroupSection and PlacementSection

Certain types of applications require the ability to specify that two or more VirtualSystems should be deployed closely together since they rely on very fast communication or a common hardware dependency such as a reliable communication link. Other types of applications require the ability to specify that two or more VirtualSystems should be deployed apart due to high-availability or disaster recovery considerations.

PlacementGroupSection allow an OVF package author to define a placement policy for a group of VirtualSystems, while PlacementSection allow the author to annotate elements with membership of a particular placement policy group.

Zero or more PlacementGroupSections may be declared at the Envelope level, while PlacementSection may be declared at the VirtualSystem or VirtualSystemCollection level, but not at both. Declaring a VirtualSystemCollection member of a placement policy group applies transitively to all child VirtualSystem elements. A VirtualSystem shall be member of at most one placement policy group.

The ovf:id attribute on PlacementGroupSection is used to identify the particular placement policy; the attribute value must be unique within the OVF package. Placement policy group membership is specified using the ovf:group attribute on PlacementSection; the attribute value must match the value of an ovf:id attribute on a PlacementGroupSection.

This version of the specification defines the placement policies "affinity" and "availability", specified with the required ovf:policy attribute on PlacementGroupSection.

Placement policy "affinity" describe that VirtualSystems should be placed as closely together as possible. The deployment platform should attempt to keep these virtual machines located as adjacentlly as possible, typically on the same physical host or with fast network connectivity between hosts.

Placement policy "availability" describe that VirtualSystems should be placed separately. The deployment platform should attempt to keep these virtual machines located apart, typically on the different physical hosts.

EXAMPLE:

<Envelope ...>
...<ovf:PlacementGroupSection ovf:id="web" ovf:policy="availability">
  <Info>Placement policy for group of VMs</Info>
  <ovf:Description>Placement policy for web tier</ovf:Description>
</ovf:PlacementGroupSection>
...
<VirtualSystemCollection ovf:id="web-tier">
  ...
  <ovf:ScaleOutSection ovf:id="web-node">
    <Info>Web tier</Info>
  </ovf:ScaleOutSection>
...
<VirtualSystem ovf:id="web-node">
  <Info>Web server</Info>
</VirtualSystem>

In the example above, all virtual machines in the compute tier should be placed separately for high availability. This example also use the ScaleOutSection defined in clause 9.14, in which case each replica get the policy assigned.

9.16 Encryption Section

For licensing and other reasons it is desirable to have an encryption scheme enabling free exchange of OVF appliances while ensuring that only the intended parties can use them. The XML Encryption Syntax and Processing standard is utilized to encrypt either the files in the reference section or any parts of the XML markup of an OVF document.

The various aspects of OVF encryption are as shown below:

1. block encryption
   The OVF document author shall utilize block encryption algorithms as specified in the XML encryption 1.1 documents (ref) for this purpose.

2. key derivation
   The OVF author may use the appropriate key for this purpose. If the key is derived using a passphrase then the author shall use one of the key derivations specified in the XML Encryption 1.1 standard.

3. Key transport.
   If the encryption key is embedded in the OVF document, the specified key transport mechanisms shall be used.

This specification defines a new section called the EncryptionSection as a focal point for the encryption functionality. This new section 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.

Note that depending on which parts of the OVF markup has been encrypted, an OVF descriptor may not validate against the OVF schemas until decrypted.
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.

EXAMPLE:

```xml
<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>
    ...
  </xenc11:DerivedKey>
  <!-- The ReferenceList element below contains references to the file Ref-109.vhd via the URI syntax which is specified by XML Encryption. -->
  <xenc:ReferenceList>
    <xenc:DataReference URI="#first.vhd"/>
    <xenc:DataReference URI="#"/>
    <xenc:ReferenceList>
      <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>
        ...
      </xenc11:DerivedKey>
      <!-- Method#2: The following example illustrates use of a symmetric key transported 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>
            ...
          </xenc:ReferenceList>
          <xenc:DataReference URI="#second-xml-fragment" to the XML fragment that has been encrypted using the above method -->
          <xenc:ReferenceList>
            <xenc:DataReference URI="#second-xml-fragment"/>
            <xenc:DataReference URI="#"/>
          </xenc:ReferenceList>
        </xenc:EncryptedKey>
      </xenc:ReferenceList>
    </xenc:DataReference URI="#second-xml-fragment"/>
    <xenc:DataReference URI="#"/>
  </xenc:ReferenceList>
</ovf:EncryptionSection>
```
Below is an example of the encrypted file which is referenced in the EncryptionSection above using URI='Ref-109.vhd' syntax.

```xml
  <!-- 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 back pointer 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>
```

Below is an example of the encrypted OVF markup which is referenced in the EncryptionSection above using URI='#second-xml-fragment' syntax.

```xml
  <!-- the EncryptedData element below encompasses encrypted xml from the original document. It is provided with the Id "first-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>
```

## 10 Internationalization

The following elements support localizable messages using the optional `ovf:msgid` attribute:

- Info element on Content
- Name element on Content
- Info element on Section
- Annotation element on AnnotationSection
- License element on EulaSection
- Description element on NetworkSection
- Description element on OperatingSystemSection
- Description, Product, Vendor, Label, and Category elements on ProductSection
Description and Label elements on Property

Description and Label elements on DeploymentOptionSection

ElementName, Caption and Description subelements on the System element in VirtualHardwareSection

ElementName, Caption and Description subelements on Item elements in VirtualHardwareSection

ElementName, Caption and Description subelements on Item elements in ResourceAllocationSection

The ovf:msgid attribute contains an identifier that refers to a message that may have different values in different locales.

EXAMPLE 1:

```
<Info ovf:msgid="info.text">Default info.text value if no locale is set or no locale match</Info>
```

```
<License ovf:msgid="license.tomcat-6_0"/>
```

The xml:lang attribute on the Envelope element shall specify the default locale for messages in the descriptor. The attribute is optional with a default value of "en-US".

10.1 Internal Resource Bundles

Message resource bundles can be internal or external to the OVF descriptor. Internal resource bundles are represented as Strings elements at the end of the Envelope element.

EXAMPLE 2:

```
<ovf:Envelope xml:lang="en-US">
...  
...  
...  
...  
<Info 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>
</ovf:Envelope>
```

10.2 External Resource Bundles

External resource bundles shall be listed first in the References section and referred to from Strings elements. An external message bundle follows the same schema as the embedded one. Exactly one Strings element shall be present in an external message bundle, and that Strings element may not have an ovf:fileRef attribute specified.

EXAMPLE 3:

```
<ovf:Envelope xml:lang="en-US">
<References>
...  
</ovf:Envelope>
```
EXAMPLE 4: Example content of external resources/it-it-bundle.msg file, which is referenced in previous example:

```xml
<Strings
  xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1"
  xml:lang="it-IT"
>
  <Msg ovf:msgid="info.os">Sistema operativo</Msg>
  ...
</Strings>
```

The embedded and external Strings elements may be interleaved, but they shall be placed at the end of the Envelope element. If multiple occurrences of a msg:id attribute with a given locale occur, a latter value overwrites a former.

10.3 Message Content in External File

Starting with version 2.0 of this specification, the content of all localizable messages may be stored in an external file using the optional ovf:fileRef attribute on the Msg element. For the License element on EulaSection in particular, this allows inclusion of a standard license text file in unaltered form without any XML header or footer.

The ovf:fileRef attribute denotes the message content by identifying an existing 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 content of an external file referenced using ovf:fileRef shall be interpreted as plain text in UTF-8 Unicode.

If the referenced file is not found, the embedded content of the Msg element shall be used.

The optional ovf:fileRef attribute may appear on Msg elements in both internal and external Strings resource bundles.

EXAMPLE 5:

```xml
<Envelop 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>
    ...
  </VirtualSystem>
  ...<Strings xml:lang="en-US">
    <Msg ovf:msgid="license" ovf:fileRef="license-en-US">Invalid license</Msg>
  </Strings>
  <Strings xml:lang="de-DE">
```
In the example above, the default license agreement is stored in plain text file license-en-US.txt, while the license agreement for the de-DE locale is stored in file license-de-DE.txt.

Note that the above mechanism works for all localizable elements and not just License.

11 OVF Environment

The OVF environment defines how the guest software and the deployment platform interact. This environment allows the guest software to access information about the deployment platform, such as the user-specified values for the properties defined in the OVF descriptor.

The environment specification is split into a protocol part and a transport part. The protocol part defines the format and semantics of an XML document that can be made accessible to the guest software. The transport part defines how the information is communicated between the deployment platform and the guest software.

The dsp8027_1.1.0.xsd XML schema definition file for the OVF environment contains the elements and attributes.

11.1 Environment Document

The environment document is an extensible XML document that is provided to the guest software about the environment in which it is being executed. The way that the document is obtained depends on the transport type.

EXAMPLE: An example of the structure of the OVF environment document follows:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Environment xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ovfenv="http://schemas.dmtf.org/ovf/environment/1"
  xmlns="http://schemas.dmtf.org/ovf/environment/1"
  ovfenv:id="identification of VM from OVF descriptor">
  <!-- Information about virtualization platform -->
  <PlatformSection>
    <Kind>Type of virtualization platform</Kind>
    <Version>Version of virtualization platform</Version>
    <Vendor>Vendor of virtualization platform</Vendor>
    <Locale>Language and country code</Locale>
    <TimeZone>Current timezone offset in minutes from UTC</TimeZone>
  </PlatformSection>
  <!-- Properties defined for this virtual machine -->
  <PropertySection>
    <Property ovfenv:key="key" ovfenv:value="value">
      <!-- More properties -->
    </Property>
  </PropertySection>
  <!-- Properties from sibling -->
  <Entity ovfenv:id="id of sibling virtual system or virtual system collection">
    <!-- Properties from sibling -->
  </Entity>
</Environment>
```
The value of the ovfenv:id attribute of the Environment element shall match the value of the ovf:id attribute of the VirtualSystem entity describing this virtual machine.

The PlatformSection element contains optional information provided by the deployment platform. Elements Kind, Version, and Vendor describe deployment platform vendor details; these elements are experimental. Elements Locale and TimeZone describe the current locale and time zone; these elements are experimental.

The PropertySection element contains Property elements with key/value pairs corresponding to all properties specified in the OVF descriptor for the current virtual machine, as well as properties specified for the immediate parent VirtualSystemCollection, if one exists. The environment presents properties as a simple list to make it easy for applications to parse. Furthermore, the single list format supports the override semantics where a property on a VirtualSystem may override one defined on a parent VirtualSystemCollection. The overridden property shall not be in the list. Overriding may occur if a property in the current virtual machine and a property in the parent VirtualSystemCollection has identical ovf:key, ovf:class, and ovf:instance attribute values; see 9.5. In this case, the value of an overridden parent property may be obtained by adding a differently named child property referencing the parent property with a macro; see 9.5.

An Entity element shall exist for each sibling VirtualSystem and VirtualSystemCollection, if any are present. The value of the ovfenv:id attribute of the Entity element shall match the value of the ovf:id attribute of the sibling entity. The Entity elements contain the property key/value pairs in the sibling’s OVF environment documents, so the content of an Entity element for a particular sibling shall contain the exact PropertySection seen by that sibling. This information can be used, for example, to make configuration information such as IP addresses available to VirtualSystems being part of a multi-tiered application.

Table 8 shows the core sections that are defined.

Table 8 – Core Sections

<table>
<thead>
<tr>
<th>Section</th>
<th>Location</th>
<th>Multiplicity</th>
</tr>
</thead>
<tbody>
<tr>
<td>PlatformSection</td>
<td>Environment</td>
<td>Zero or one</td>
</tr>
<tr>
<td>Provides information from the deployment platform</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PropertySection</td>
<td>Environment</td>
<td>Zero or one</td>
</tr>
<tr>
<td>Contains key/value pairs corresponding to properties defined in the OVF descriptor</td>
<td>Entity</td>
<td></td>
</tr>
</tbody>
</table>

The environment document is extensible by providing new section types. A consumer of the document should ignore unknown section types and elements.

11.2 Transport

The environment document information can be communicated in a number of ways to the guest software. These ways are called transport types. The transport types are specified in the OVF descriptor by the ovf:transport attribute of VirtualHardwareSection. Several transport types may be specified, separated by a single space character, in which case an implementation is free to use any of them. The transport types define methods by which the environment document is communicated from the deployment platform to the guest software.

To enable interoperability, this specification defines an "iso" transport type which all implementations that support CD-ROM devices are required to support. The iso transport communicates the environment document by making a dynamically generated ISO image available to the guest software. To support the iso transport type, prior to booting a virtual machine, an implementation shall make an ISO read-only
disk image available as backing for a disconnected CD-ROM. If the iso transport is selected for a VirtualHardwareSection, at least one disconnected CD-ROM device shall be present in this section.

The generated ISO image shall comply with the ISO 9660 specification with support for Joliet extensions.

The ISO image shall contain the OVF environment for this particular virtual machine, and the environment shall be present in an XML file named ovf-env.xml that is contained in the root directory of the ISO image. The guest software can now access the information using standard guest operating system tools.

If the virtual machine prior to booting had more than one disconnected CD-ROM, the guest software may have to scan connected CD-ROM devices in order to locate the ISO image containing the ovf-env.xml file.

The ISO image containing the OVF environment shall be made available to the guest software on every boot of the virtual machine.

Support for the "iso" transport type is not a requirement for virtual hardware architectures or guest operating systems which do not have CD-ROM device support.

To be compliant with this specification, any transport format other than iso shall be given by a URI which identifies an unencumbered specification on how to use the transport. The specification need not be machine readable, but it shall be static and unique so that it may be used as a key by software reading an OVF descriptor to uniquely determine the format. The specification shall be sufficient for a skilled person to properly interpret the transport mechanism for implementing the protocols. It is recommended that these URIs are resolvable.
ANNEX A
(informative)

Symbols and Conventions

XML examples use the XML namespace prefixes defined in Table 1. The XML examples use a style to not specify namespace prefixes on child elements. Note that XML rules define that child elements specified without namespace prefix are from the namespace of the parent element, and not from the default namespace of the XML document. Throughout the document, whitespace within XML element values is used for readability. In practice, a service can accept and strip leading and trailing whitespace within element values as if whitespace had not been used.

Syntax definitions in this document use Augmented BNF (ABNF) as defined in IETF RFC5234 with the following exceptions:

- Rules separated by a bar (|) represent choices, instead of using a forward slash (/) as defined in ABNF.
- Any characters must be processed case sensitively, instead of case-insensitively as defined in ABNF.
- Whitespace (i.e., the space character U+0020 and the tab character U+0009) is allowed between syntactical elements, instead of assembling elements without whitespace as defined in ABNF.
## ANNEX B
(informative)

### Change Log

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0.0</td>
<td>2009-02-22</td>
<td>DMTF Standard</td>
</tr>
<tr>
<td>1.1.0</td>
<td>2010-01-12</td>
<td>DMTF Standard</td>
</tr>
<tr>
<td>1.1.1</td>
<td>2010-06-01</td>
<td>Incorporate ANSI editor changes (wgv0.5.0)</td>
</tr>
<tr>
<td></td>
<td>2010-06-23</td>
<td>Address Mantis 0000691 (wgv0.5.1)</td>
</tr>
<tr>
<td></td>
<td>2010-06-24</td>
<td>Update POSIX reference to <a href="https://www.ieee.org">ISO/IEC/IEEE 9945:2009</a> (wgv0.6.0)</td>
</tr>
<tr>
<td>2.0.0a</td>
<td>wgv 0.7.0</td>
<td>Work in Progress release</td>
</tr>
<tr>
<td>2.0.0b</td>
<td>wgv 0.9.0</td>
<td>Work in Progress release candidate - Result of F2F, incorporated review comments, moved to Word 2010 &amp; new template</td>
</tr>
<tr>
<td>2.0.0b</td>
<td>wgv 0.9.1</td>
<td>Work in Progress release candidate - Result of WG ballot</td>
</tr>
<tr>
<td></td>
<td>2011-12-14</td>
<td>Change 10.6 to Shishir's material, update list of contributors, added XML encryption to normative references</td>
</tr>
<tr>
<td>2.0.0c</td>
<td>wgv 0.9.2</td>
<td>NetworkSection and VirtualHardwareSection related section changes based on OVF 2 schema changes for network port profiles.</td>
</tr>
<tr>
<td>2.0.0c</td>
<td>wgv 0.9.3</td>
<td>Specs changes to reflect the new definitions of NetworkSection, VirtualHardwareSection, and ResourceAllocationSection.</td>
</tr>
</tbody>
</table>
ANNEX C
(normative)

OVF XSD

Normative copies of the XML schemas for this specification may be retrieved by resolving the following URLs:

- http://schemas.dmtf.org/ovf/envelope/1/dsp8023_1.2.0.xsd
- http://schemas.dmtf.org/ovf/envelope/2/dsp8023_2.0.0.xsd
- http://schemas.dmtf.org/ovf/environment/1/dsp8027_1.1.0.xsd

Any xs:documentation content in XML schemas for this specification is informative and provided only for convenience.

Normative copies of the XML schemas for the WS-CIM mapping (DSP0230) of CIM_ResourceAllocationSystemSettingsData and CIM_VirtualSystemSettingData may be retrieved by resolving the following URLs:

- http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_EthernetPortAllocationSettingData.xsd
- http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_StorageAllocationSettingData.xsd

This specification is based on the following CIM MOFs:

- CIM_VirtualSystemSettingData.mof
- CIM_ResourceAllocationSettingData.mof
- CIM_EthernetPortAllocationSettingData.mof
- CIM_StorageAllocationSettingData.mof
- CIM_OperatingSystem.mof
ANNEX D
(informative)

OVF Mime Type Registration Template

Registration Template
To: ietf-types@iana.org
Subject: Registration of media type Application/OVF
Type name: Application
Subtype name: OVF
Required parameters: none
Optional parameters: none
Encoding considerations: binary

Security considerations:
• An OVF package contains active content that is expected to be launched in a virtual machine.
The OVF standard, section 5.1 states: “An OVF package may be signed by signing the manifest file. The digest of the manifest file is stored in a certificate file with extension .cert file along with the base64-encoded X.509 certificate. The .cert file shall have the same base name as the .ovf file and be a sibling of the .ovf file. A consumer of the OVF package shall verify the signature and should validate the certificate.

• An OVF package may contain passwords as part of the configuration information. The OVF standard, section 9.5 states: “The optional Boolean attribute ovf:password indicates that the property value may contain sensitive information. The default value is FALSE. OVF implementations prompting for property values are advised to obscure these values when ovf:password is set to TRUE. This is similar to HTML text input of type password. Note that this mechanism affords limited security protection only. Although sensitive values are masked from casual observers, default values in the OVF descriptor and assigned values in the OVF environment are still passed in clear text.”

Interoperability considerations:
• OVF has demonstrated interoperability via multiple, interoperating implementations in the market.

Published specification:
• DSP0243_2.0.0.pdf

Applications that use this media type:
• Implementations of the DMTF Standard: Cloud Infrastructure Management Interface (CIMI) (DSP0263_1.0.0.pdf)
• Implementations of the SNIA Cloud Data Management Interface (CDMI) – OVF Extension
Additional information:
- Magic number(s): none
- File extension(s): .ova
- Macintosh file type code(s): none
- Person & email address to contact for further information:
- Intended usage: (One of COMMON, LIMITED USE or OBSOLETE.)
- Restrictions on usage: (Any restrictions on where the media type can be used go here.)
- Author:
- Change controller:
Bibliography

2096 ISO 9660, Joliet Extensions Specification, May 1995,
2097 http://bmrc.berkeley.edu/people/chaffee/jolspec.html

2098 W3C, Best Practices for XML Internationalization, October 2008,
2099 http://www.w3.org/TR/2008/NOTE-xml-i18n-bp-20080213/

2100 DMTF DSP1044, Processor Device Resource Virtualization Profile 1.0
2101 http://www.dmtf.org/standards/published_documents/DSP1044_1.0.pdf

2102 DMTF DSP1045, Memory Resource Virtualization Profile 1.0

2104 DMTF DSP1047, Storage Resource Virtualization Profile 1.0
2105 http://www.dmtf.org/standards/published_documents/DSP1047_1.0.pdf

2106 DMTF DSP1022, CPU Profile 1.0,
2107 http://www.dmtf.org/standards/published_documents/DSP1022_1.0.pdf

2108 DMTF DSP1026, System Memory Profile 1.0,
2109 http://www.dmtf.org/standards/published_documents/DSP1026_1.0.pdf

2110 DMTF DSP1014, Ethernet Port Profile 1.0,
2111 http://www.dmtf.org/standards/published_documents/DSP1014_1.0.pdf

2112 DMTF DSP1050, Ethernet Port Resource Virtualization Profile 1.1

2114 DMTF DSP8049, Network Port Profile XML Schema 1.0
2115 http://schema.dmtf.org/ovf/networkportprofile/1/DSP8049_1.0.xsd

2116