OVF Tool User's Guide

Open Virtualization Format Tool 4.3.0

vmware®

You can find the most up-to-date technical documentation on the VMware website at:

https://docs.vmware.com/

If you have comments about this documentation, submit your feedback to

docfeedback@vmware.com

VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

Contents

About This Book

This *OVF Tool User's Guide* provides information about how to use VMware® OVF Tool to package virtual machines and vApps into Open Virtualization Format (OVF) standard packages.

Revision History

A revision occurs with each release of the product, or as needed. A revised version can contain minor or major changes. Table 1 lists the versions of this manual.

Table 1. Revision History

Revision	Description
05/2018	OVF Tool 4.3.0 User's Guide. Large upload retry, SHA digest fix, NVM and PMEM, Curl update, better logging, VBS and TPM support.
10/2016	OVF Tool 4.2.0 User's Guide. New options for SSL version and cipher list, NVRAM support for EFI boot.
02/2016	Mention need for explicit extraConfig flags on ESXi hosts.
03/2015	OVF Tool 4.1.0 User's Guide. Added DVS port group, `viUseProxy` option for vSphere, and vCloud locators.
12/2014	Increased Open SSL compatibility version to 1.0.1j. See http://www.openssl.org/news/vulnerabilities.html.
10/2014	OVF Tool 4.0.0 User's Guide. Added --allowAllExtraConfig and --decodeBase64 command line options.
06/2014	OVF Tool 3.5.2 User's Guide. Increased security for Open SSL.
04/2014	OVF Tool 3.5.1 User's Guide.
08/2013	OVF Tool 3.5.0 User's Guide. Includes new command line options.
08/2012	OVF Tool 3.0.1 User's Guide.
08/2011	OVF Tool 2.1 User's Guide.
06/2010	OVF Tool 2.0.1 Guide.
05/2009	OVF Tool 1.0 Guide.

Intended Audience

This book is intended for anyone who needs to convert an OVF package to a virtual machine, or a virtual machine to an OVF package. Users typically include: system administrators, software developers, QA engineers, and anyone who need to package or unpackage virtual machines using open industry standards.

Document Feedback

VMware welcomes your suggestions for improving our documentation. If you have comments, send your feedback to docfeedback@vmware.com.

Technical Support and Education Resources

The following sections describe the technical support resources available to you. To access the current version of this book and other books, go to http://www.vmware.com/support/pubs.

OVF Tool Users' Forum

To obtain more information and to post questions about OVF Tool, go to the OVF Tool Forum at http://www.vmware.com/go/ovftool.

Support Offerings

To find out how VMware support offerings can help meet your business needs, go to http://www.vmware.com/support/services.

Overview of the OVF Tool

1

Open Virtualization Format (OVF) is an industry standard to describe metadata about virtual machine images in XML format. VMware OVF Tool is a command-line utility that helps users import and export OVF packages to and from a wide variety of VMware products.

This chapter includes the following topics:

The Open Virtualization Format

VMware implemented a tool for importing and exporting virtual machines in OVF standard format.

The OVF Standard

The OVF specification describes a secure, portable, efficient, and flexible method to package and distribute virtual machines and components. It originated from the Distributed Management Task Force (DMTF) after vendor initiative. Companies that contributed to the OVF standard include VMware, Dell, HP, IBM, Microsoft, XenSource, and Citrix.

Version 1.1 was published in January 2010, which supersedes the 1.0 specification published April 2009, and is available on the DMTF Web site, along with a white paper.

- Specification: http://www.dmtf.org/standards/published_documents/DSP0243_1.1.0.pdf
- Whitepaper: http://www.dmtf.org/standards/published_documents/DSP2017_1.0.0.pdf

Benefits of OVF

Using OVF to distribute virtual machines has the following benefits:

- Ease of use. When users receive a package in OVF format, they do not have to unzip files, execute binaries, or convert disk formats. Adding a vApp can be as simple as typing a URL and clicking Install.

- Virtual hardware validation. OVF supports fast and robust hardware validation. You do not have to install a complete virtual machine before determining whether it is compatible with an ESXi host (for example, because it uses IDE virtual disks).
- Metadata inclusion. Additional metadata, such as an end-user license agreement, can be packaged with the OVF and displayed before installation.
- Optimized download from the Internet. Large virtual disks are compressed for fast download and to reduce disk space for large template libraries.

Using the VMware OVF Tool

The VMware OVF Tool is available on many platforms and can be downloaded if not built into a product.

VMware Platforms Using the OVF Standard

VMware supports the OVF standard on the following platforms:

- Use the OVF Tool 3.x for vSphere 4.0 and later, vCloud Director 1.5, 5.1, and 5.5, vCloud Director 1.0 (for OVF and OVA types only), vCenter 2.5 and later, ESX 3.5 and later, VMware Server 2, VMware Workstation 6.0 and later, and VMware Fusion 3.0 and later.
- OVF 0.9 is supported for import and export by VirtualCenter 2.5 and later, and ESX 3.5 and later.
- VMware Studio 1.0 and later can generate OVF packages.

OVF support is built into the vSphere Client that installs from, and is compatible with vCenter 5.0 and ESXi 5.0, vCenter 4.0 and ESX 4.0. It is also built into the vSphere Client that installs from and is compatible with VirtualCenter 2.5 and later, and ESX 3.5 and later. The vSphere 5.1 Web Client includes the 3.x version of the VMware OVF Tool as part of the Client Integration Plug-in.

Setting Up the OVF Tool

You can find the latest information about System Requirements, supported VMware software and platforms, installation, and known issues by reading the latest release notes located at the following web page: www.vmware.com/support/developer/ovf.

OVF Tool Highlights

The OVF Tool provides the following key features:

- Supports import and generation of OVA packages (OVA is part of the OVF standard, and contains all the files of a virtual machine or vApp in a single file.)
- Directly converts between any vSphere, vCloud Director, VMX, or OVF source format to any vSphere, vCloud Director, VMX, or OVF target format
- Accesses OVF sources using HTTP, HTTPS, FTP, or from a local file
- Deploys and exports vApp configurations on vSphere 4.0 (and all newer) targets and on vCloud Director 1.5 (and all newer) targets

- Provides options to power on a VM or vApp after deployment, and to power off a virtual machine or vApp before exporting (caution advised)
- Show information about the content of any source in probe mode
- Provides context sensitive error messages for vSphere and vCloud Director sources and targets, showing possible completions for common errors, such as an incomplete vCenter inventory path or missing datastore and network mappings
- Provides an optional output format to support scripting when another program calls OVF Tool
- Uses new optimized upload and download API (optimized for vSphere 4.0 and newer)
- Signs OVF packages and validates OVF package signatures
- Validates XML Schema of OVF 1.0 and OVF 1.1 descriptors
- Import and export of OVF packages into a vApprun 1.0 workspace.

  For more information about vApprun, see http://labs.vmware.com/flings/vapprun.

Examples of OVF Tool Use

OVF import and export are built into the vSphere Client and the vSphere Web Client, and various other enterprise oriented VMware products.

For VMware products without built-in OVF support, or when you need to accomplish specialized OVF operations, you can download the OVF Tool over the Web.

Importing and Exporting OVF Using vSphere Client

Using the vSphere 5.0 or 4.0 Client, you can import an OVF package and export a vApp into an OVF package. For example, to import an OVF package using vSphere Client 4:

Click **File > Deploy OVF Template**.

For example, to export a vApp into an OVF package using vSphere Client 4:

Click **File > Export OVF Template** > **Export**.

Using the vSphere Client 2.5, you can import an OVF virtual machine into an ESXi host and export a virtual machine to an OVF file (note that vSphere Client 2.5 is limited to OVF 0.9). For example, to import an OVF vApp into an ESXi host using vSphere Client 2.5:

Click **File > Virtual Appliance > Import** .

For example, to export a virtual machine to an OVF file using vSphere Client 2.5:

Click **File > Virtual Appliance > Export** .

OVF packages imported or exported by OVF Tool are completely compatible with packages imported or exported by the vSphere Client or the vSphere Client.

VMware OVF Tool Delta Disk Facilities

VMware OVF Tool automatically compresses disk files. In the streaming VMDK files that OVF Tool generates, the tool compresses each 64KB disk grain. It is possible to achieve even better compression using the `--compress` option. In addition, if a package contains multiple virtual machines, it is possible to compress an OVF package even more using a technique called delta disk compression. This compression algorithm is invoked using the `--makeDeltaDisks` option.

```
ovftool --makeDeltaDisks package.ovf output-dir/
```

Delta disk compression identifies disk segments that are equal and combines these equal parts in a parent disk. This process prevents storing the same segment twice.

As an example, consider a software solution that consists of an Apache Web server virtual machine and a MySQL database virtual machine, both installed on top of a single-disk Ubuntu server. The two virtual machines were created with the following process:

1 Create a plain Ubuntu installation on one virtual machine.

2 Clone the virtual machine.

3 Install Apache on the first virtual machine.

4 Install MySQL on the second virtual machine.

Using delta disk compression on the two virtual machine disks creates a parent disk containing all of the information they share, which is essentially the entire operation system and two child disks containing the MySQL and Apache parts.

A plain Ubuntu server can use 400–500MB of space, and two would use 800–1000MB of space. By contrast, using delta disk compression, an OVF package with these two servers uses only 400–500MB (plus the size of the MySQL and Apache installations), which saves 400–500MB by not duplicating the Ubuntu server.

Any number of disks can be combined creating various disk trees and saving more space.

vSphere 4 and later support the deployment of OVF packages that contain delta disk hierarchies.

For delta disk compression, keep in mind the following:

- Only disks with equal capacity can be combined. If you expect to use delta disk compression, you must keep disk capacities equal.
- Delta disk compression necessitates that segments that might be put in a parent disk are at the same offset from the beginning of their respective files. In the Ubuntu example, if the setup varies between the two installations, it can completely offset each segment on one of the disks from the segments on the other disk. In this case, delta disk compression does not produce any significant disk space savings. This is why the example specified cloning the Ubuntu server before installing the MySQL and Apache parts, respectively.
- Delta disk compression takes OVF packages and vSphere and VMX files as input, but not OVA packages.

- The delta disk compression algorithm needs to read the contents of each disk up to two times. It might make sense to invoke OVF Tool on a local copy of the OVF package.
- The delta disk compression algorithm always generates an OVF package in the given output directory. To convert this OVF package into an OVA package, reinvoke OVF Tool.

Using the VMware OVF Tool 2

The VMware OVF Tool is a command-line utility that supports importing and exporting of OVF packages, VMX files, or virtual machines from ESXi hosts and other VMware products.

This chapter includes the following topics:

Definitions and Command Syntax

A source location or source URL locator refers to an OVF package, VMX file, or virtual machine in a VMware product, such as VMware Workstation, vSphere ESXi Host, vSphere vCenter Server, vCloud Director, or vFabric Data Director.

A target location or destination URL locator specifies either a file location, or a location within a VMware product, such as VMware Workstation, ESXi, vCenter Server, vCloud Director or vFabric Data Director.

Run VMware OVF Tool From the Command Line

Procedure

1 At the command-line prompt, run the OVF Tool as follows

```
ovftool <source locator> <target locator>
```

`<source locator>` and `<target locator>` are paths to the source and target for the virtual machine, OVF package, OVA package, or vSphere location. See Command-Line Options for options. Table 2-1 describes the source and target locators. For details, see Specifying a Locator.

If you are using an operating system where spaces are not allowed in paths on the command line, and need the full path to run OVF Tool, enclose the path in quotes as shown below:

```
"/Applications/VMware OVF Tool/ovftool" --help
```

2 If you want to specify additional options, type them before the source and target locators.

```
ovftool <options> <source locator> <target locator>
```

3 To display all options, type **`ovftool –h`**.

Probe mode allows you to investigate the contents of a source. To invoke probe mode, use the `ovftool` command with only a source and no target. OVF Tool prints information about the source such as hardware, EULA, and OVF properties.

```
ovftool <options> <source locator>
```

Use probe mode to examine an OVF package before deploying it. For example, you can examine the download and deployment sizes, determine the set of networks to be mapped, determine the OVF properties to be configured, read the EULA, and determine the virtual hardware requirements.

The probe operation is fast, as it only needs to access the OVF descriptor. It does not need to download the entire OVA or VMDK files. Probe mode also validates the certificate if the source is signed. For details about Probe Mode and sample output, see Chapter 5 Using the VMware OVF Tool Probe Mode.

Run OVF Tool from ESXi instead of vCenter

If you are deploying with the `ovftool` command from vCenter Server, you can use the command line options in the table below to specify parameters when you deploy.

If you are deploying with the `ovftool` command from an ESXi host, you must "inject" the parameters into the resulting VM when it is powered on. This is because the ESXi host lacks a cache to store the OVF parameters, as in vCenter Server. Therefore, you must use the `--X:injectOvfEnv` debug option with the `--poweron` flag in the command line if you are deploying a virtual machine from ESXi. Example below. (You can also do this using the Create a VM from an OVA/OVF option in the ESXi host client, then browse to the .ova file.)

```
>./ovftool/ovftool\
--name="Cloudvm_2074586_with_inject"\
--X:injectOvfEnv\
--X:logFile=ovftool.log\
--X:logLevel=trivia\
--acceptAllEulas\
-ds=cl-storage-1\
-dm=thin\
--net:'Network 1=VM Network'\
--X:enableHiddenProperties\
--noSSLVerify\
--allowExtraConfig\
--machineOutput\
--prop:vami.netmask0.VMware_vCenter_Server_Appliance=255.255.255.0\
--prop:guestinfo.cis.appliance.net.prefix=24\
--prop:guestinfo.cis.appliance.net.gateway=X.X.X.123\
--prop:guestinfo.cis.appliance.time.tools-sync=True\
```

```
--prop:vami.gateway.VMware_vCenter_Server_Appliance=X.X.X.123\
--prop:guestinfo.cis.appliance.net.dns.servers=X.X.X.1,X.X.X.2\
--prop:vami.ip0.VMware_vCenter_Server_Appliance=X.X.X.145\
--prop:guestinfo.cis.appliance.root.passwd=vmware\
--prop:guestinfo.cis.appliance.net.addr=X.X.X.145\
--prop:vami.DNS.VMware_vCenter_Server_Appliance=X.X.X.1,X.X.X.2\
--prop:vami.vmname=vmc-srm-vc10\
--prop:guestinfo.cis.appliance.root.shell=/bin/bash\
--prop:guestinfo.cis.vmdir.first-instance=True\
--prop:guestinfo.cis.appliance.ssh.enabled=True\
--prop:guestinfo.cis.appliance.net.mode=static\
--prop:guestinfo.cis.appliance.net.addr.family=ipv4\
--prop:guestinfo.cis.vmdir.domain-name=vsphere.local\
--prop:guestinfo.cis.vmdir.password=vmware\
--powerOn\
--X:waitForIp\
http://<directory_w/cloudvm>/VMware-vCenter-Server-Appliance-6.0.0.XXXX-XXXXXXX_OVF10.ovf\
vi://root:pwd@vm_name
```

You will need to replace the variables (IP addresses, build numbers, root password, and VM names) in the above example with values from your own system.

Deploy a Virtual Machine to Static DVS Port Group

At this time, you can not use the OVF Tool to deploy a virtual machine to a static port group.

To work around this issue:

Procedure

1. Use vCenter Server to create an ephemeral port on the desired network.
2. Deploy the appliance to that port group on an ESXi host.
3. Switch the appliance over to that static port group.
4. Use vCenter Server to delete the ephemeral port group.

Command-Line Options

For every command, you specify the source and target locators. Table 2-1 defines each locator type.

Table 2-1. OVF Tool Definitions of Source and Target Locators

Locator	Definition
`<source locator>`	Path to the source, which must be either a virtual machine, vApp, vApprun workspace entity, or an OVF package. The source locator can be one of the following: ■ A path to an OVF or OVA file (a local file path, or an HTTP, HTTPS, or FTP URL). ■ A virtual machine (a local file path to a .vmx file). ■ A vSphere locator identifying a virtual machine or vApp on vCenter, ESXi, or VMware Server. ■ A vCloud Director locator identifying a virtual machine or a vApp in vCloud Director. ■ A local file path to a vApprun workspace entity.
`<target locator>`	The target locator can be one of the following: ■ A local file path for VMX, OVF, OVA, or vApprun workspace. ■ A vSphere locator identifying a cluster, host, or a vSphere location. ■ A vCloud Director locator identifying a virtual machine or a vApp in vCloud Director.

Table 2-2 shows all the command-line options.

Options perform actions only between certain source and target types. Table 2-2 shows which source and target types each option works with. If you specify an option using an irrelevant source or target type, the command does nothing.

All options can be set using the form `--option=value`.

Binary options can be enabled or disabled explicitly. For example: `--option=true`, `--option=false`.

Table 2-2. OVF Tool Command-Line Options

Option's Long Name	Short Name	Source Types	Target Types	Description
`--acceptAllEulas`		OVF, OVA	N/A	Accepts all end-user licenses agreements (EULAs) without being prompted. Binary option.
`--allowExtraConfig`				Lets you specify the extra config options that can be converted to `.vmx` format. These options are a security risk as they control low-level and potential unsafe options on the VM. Each option must be specified using a series of key value pairs (sometimes referred to as a white list).
`--allowAllExtraConfig`				Converts all extra config options to `.vmx` format. Flags must be explicitly specified on ESXi hosts, but with vCenter Server `--exportflags=extraconfig` produces vmw:ExtraConfig.
`--annotation`		All		Adds annotation to vi, vmx, vapprun, vCloud, OVF, and OVA source locators.
`--authdPortSource`		vSphere	vSphere	Overrides default VMware `authd` port (902) when using a host as source.
`--authdPortTarget`		vSphere	vSphere	Overrides the default VMware `authd` port (902) when using a host as target.

Table 2-2. OVF Tool Command-Line Options (Continued)

Option's Long Name	Short Name	Source Types	Target Types	Description
--chunkSize		N/A	OVF, OVA	Specifies the chunk size to use for files in a generated OVF or OVA package. The default is not to chunk. If you don't specify a unit for the chunk size, the chunk size is assumed to be in megabytes (mb). Accepted units are b, kb, mb, gb. Example: 2gb or 100kb. When using this option, all output files (except the OVF descriptor, manifest and certificate files) are sliced into the specified chunk size. This is useful if you need to transport an OVF package on a series of 800MB CD-ROMs, or are only able to create files up to 2GB on FAT32 file systems. When you use chunking with the OVA package option, the result is similar to OVF because all the files are chunked, but the OVA package is still be a single file.
--compress		N/A	OVF, OVA	Compresses the disk when given an OVF or OVA target locator. The value must be between 1 and 9. Use 9 for the slowest processing time, but best compression. Use 1 for the fastest processing time, but least compression.
--computerName				Sets the computer name in the guest for a VM using the syntax --computerName:<VM ID>=<value>. Only applies to vCloud targets of version 5.5 or newer.
--coresPerSocket				Specifies the distribution of the total number of CPUs over a number of virtual sockets using the syntax --coresPerSocket:<VM ID>=<value>. Only applies to vCloud targets of version 5.5 or newer.
--datastore	-ds	N/A	vSphere	Target datastore name for a vSphere locator.
--decodeBase64				Enables Base64 decoding for values in the OVFTool command line.
--defaultStorageProfile				The storage profile for all VMs in the OVF package. The value should be an SPBM profile ID. Only applies to VI targets of version 5.5 or newer.
--defaultStorageRawProfile				The storage profile for all VMs in the OVF package. The value should be a raw SPBM profile. The value overwrites that in --defaultStorageProfile. Only applies to VI targets of version 5.5 or newer
--deploymentOption		OVF, OVA	N/A	Deployment options for a deployed OVF package (if the source OVF package supports multiple options.) An OVF package can contain several deployment configurations. This option allows you to select which configuration to use when deploying to the vSphere target.
--disableVerification		OVF, OVA	N/A	Skips validation of signature and certificate. Binary option.

Table 2-2. OVF Tool Command-Line Options (Continued)

Option's Long Name	Short Name	Source Types	Target Types	Description
--diskMode	-dm	N/A	VMX, vApprun, vSphere	Select target disk format. Supported formats are: monolithicSparse, monolithicFlat, twoGbMaxExtentSparse, twoGbMaxExtentFlat, seSparse (vSphere target), eagerZeroedThick (vSphere target), thin (vSphere target), thick (vSphere target), sparse, and flat.
--diskSize				Sets the size of a VM disk in megabytes using the syntax --diskSize:<VM ID>,<disk instance ID>=<value>. Only applies to vCloud targets of version 5.5 or newer.
--eula		N/A	OVF, OVA	Inserts the EULA in the first virtual system or virtual system collection in the OVF. If the EULA is in a file, use the format: --eula@=filename
--exportFlags				Specifies the source of an export. The supported values for vSphere source are: mac, uuid, and extraconfig. The supported value for vCloud source is preserveIdentity. You can provide one or more options, separated by commas.
--extraConfig		N/A	All	Sets an ExtraConfig element for all VirtualHardwareSections. The syntax is --:extraConfig:<key>=<value> This option applies to vi, vmx, vapprun, vCloud, ovf, and ova source locators
--fencedMode		vCloud		If a parent network exists on a vCloud target, this option specifies the connectivity to the parent. Possible values are bridged, isolated, allowIn, allowInOut, allowOut.
--help	-h	N/A	N/A	Prints the VMware OVF Tool help message that lists the - help options.
--hideEula		OVF, OVA	N/A	Does not include the EULA in the OVF probe output. Binary option.
--I:morefArgs		vSphere	vSphere	Integration option. Interpret arguments for networks, datastores, and folders as VIM Managed Object Reference identifiers (type:id) for vSphere source and destination locators.
--I:sourceSessionTicket		vSphere	vSphere	Integration option. Specifies the session ticket used for authenticating the vSphere source locator.
--I:targetSessionTicket		vSphere	vSphere	Integration option. Specifies the session ticket used for authenticating the vSphere target locator.
--ipAllocationPolicy		OVF, OVA	N/A	IP allocation policy for a deployed OVF package. Supported values are: dhcpPolicy, transientPolicy, fixedPolicy, or fixedAllocatedPolicy. In OVF descriptors, you can specify a VMware specific IP assignment policy that guides the deployment process by expressing which of the policies the OVF package supports. Only values listed in the OVF descriptor are supported when the OVF or OVA package is deployed.

Table 2-2. OVF Tool Command-Line Options (Continued)

Option's Long Name	Short Name	Source Types	Target Types	Description
--ipProtocol		OVF, OVA	N/A	Specifies which IP protocol to use. For example, IPv4, IPv6. As with the ipAllocationPolicy option, you can specify which IP version this OVF package uses when it is deployed. Use only the values listed in the OVF descriptor.
--lax		OVF, OVA	N/A	Relax OVF specification conformance and virtual hardware compliance checks. (For advanced users only.)
--locale		OVF, OVA	N/A	Selects the locale for the target.
--machineOutput		N/A	N/A	Outputs OVF Tool messages in a machine readable format. Binary option.
--makeDeltaDisks		OVF, vSphere, VMX, vApprun	Must be directory	Use delta disk compression to create an OVF package from a disk source. Binary option.
--maxVirtualHardwareVersion				The maximal virtual hardware version to generate.
--memorySize				Sets the memory size in megabytes of a VM using the syntax --memorySize:<VM ID>=<value>. Example: --memorySize:vm1=1024. Applies to VI and vCloud targets of version 5.5 or newer.
--name	-n	N/A	All	Specifies the target name. Defaults to the source name.
--net		OVF, OVA	N/A	Sets a network assignment in the deployed OVF package. For example, --net:<OVF name>=<target name>. OVF packages contain symbolic names for network names which are assigned with this option. For multiple network mappings, repeat the option, separating them with a space for example, --net:s1=t1 --net:s2=t2 --net:s3=t3. If the target is vCloud 5.5 or newer, a fence mode can also be specified using the syntax --net:<OVF name>=<target name>,<fence mode>. Possible fence mode values are: bridged, isolated, and natRouted.
--network	-nw	OVF, OVA	N/A	Target network for a vSphere deployment. Use this option in place of the --net option when only one network exists in the OVF package. This option maps the symbolic OVF name to the specified network name.
--nic				Specifies NIC configuration in a VM using the syntax --nic:<VM ID>,<index>=<OVF net name>,<isPrimary>,<ipAddressingMode>,<ipAddress>. Possible values for ipAddressingMode are: DHCP, POOL, MANUAL, and NONE. ipAddress is optional and should only be used when ipAddressingMode is set to MANUAL. Only applies to vCloud targets of version 5.5 or newer.

Table 2-2. OVF Tool Command-Line Options (Continued)

Option's Long Name	Short Name	Source Types	Target Types	Description
`--noDisks`		N/A	All	Creates and uploads the virtual machine or vApps but does not upload any disk files. Disks are created empty. (Disables disk conversion.)
`--noImageFiles`		N/A	All	Creates and uploads the virtual machine or vApps but does not upload ISO files to a CD-ROM. (Does not include image files in destination.
`--noSSLVerify`				Skip SSL verification for vSphere connections.
`--numberOfCpus`				Sets the number of CPUs for a VM using the syntax `--numberOfCpus:<VM ID>=<value>`. Applies to VI and vCloud targets of version 5.5 or newer.
`--overwrite`	`-o`	N/A	All	Forces an overwrite of existing files. Binary option.
`--powerOffSource`		vCloud, vSphere	N/A	Ensures that a virtual machine or vApp is powered off before importing from a vSphere source. Binary option.
`--powerOffTarget`		N/A	vCloud, vSphere	Ensures that a virtual machine or vApp is powered off before overwriting a vSphere target. Binary option.
`--powerOn`		N/A	vCloud, vSphere	Powers on a virtual machine or vApp deployed on a vSphere target. Binary option.
`--privateKey`		N/A	OVF, OVA	Signs the OVF package with the given private key (`.pem` file). The file must contain a private key and a certificate.
`--privateKeyPassword`		N/A	OVF, OVA	Password for the private key. Used in conjunction with `--privateKey` if the private key requires password authentication. If required but not specified, the tool prompts for the password.
`--prop`		OVF, OVA	N/A	Sets a property in the deployed OVF package. For example, `--prop:<key>=<value>`. Use probe mode to learn which properties an OVF package can set. For multiple property mappings, repeat the option, separating them with a blank, for example `--prop:p1=v1 --prop:p2=v2 --prop:p3=v3`.
`--proxy`		OVF, OVA, vCloud, vSphere	OVF, OVA, vCloud, vSphere	Specifies the proxy used for HTTP, HTTPS, FTP, vSphere and vCloud access. The proxy is expressed as the URL to the proxy. For example, for `proxy.example.com`, the option value is: https://proxy.example.com:345 OVF Tool supports proxies that require authentication. If you do not provide credentials in the URL, the OVF Tool prompts you for them. You can also use the `--X:viUseProxy` option for vsphere and the `--X:vCloudUseProxy` option for vcloud but only if you use them together.
`--proxyNTLMAuth`		OVF, OVA, vCloud, vSphere	OVF, OVA, vCloud, vSphere	Enables support for the NTLM authentication and security protocol. NT Lan Manager is the authentication protocol used on networks that include systems running the Windows operating system and on stand-alone systems.

Table 2-2. OVF Tool Command-Line Options (Continued)

Option's Long Name	Short Name	Source Types	Target Types	Description
`--quiet`	`-q`	N/A	N/A	Prints only errors. No other output is sent to the screen. Binary option.
`--schemaValidate`		OVF, OVA	N/A	Validates OVF descriptor against the OVF schema. Binary option.
`--shaAlgorithm`		sha1, sha256, sha512		Use this option to condense with Secure Hash Algorithm (SHA) for manifest validation, digital signing, and OVF package creation. Either `sha1` (SHA-1), `sha256` (SHA-256), or `sha512` (SHA-512). The default value is sha256.
`--skipManifestCheck`		OVF, OVA	N/A	Skips validation of the OVF package manifest. Binary option.
`--skipManifestGeneration`		N/A	OVF, OVA	Skips generation of the OVF package manifest. Binary option.
`--sourcePEM`				File path to a Privacy Enhanced Mail (.pem) file used to verify vSphere connections. Example: --sourcePEM:<filename>.pem
`--sourceSSLThumbprint`		vSphere	N/A	SSL thumbprint of the source. OVF Tool verifies the SSL thumbprint that it receives from the source, if this value is set.
`--sourceType`	`-st`	OVF, OVA, VMX, VMX, VI, vCloud, ISO, FLP, vApprun	N/A	Explicitly expresses that the source is `OVF, OVA, VMX, VMX, vSphere, vCloud, ISO, FLP,` or `vApprun`.
`--sslCipherList`				Allows override of the default cipher list. Diffie-Hellman is disabled by default.
`--sslVersion`				Either `TLSv1_0`, `TLSv1_1`, or `TLSv1_2`. The `TLSv1_0` protocol is disabled by default.
`--storageProfile`				Sets the storage profile for a VM using the syntax `--storageProfile:<VM ID>=<value>`. Applies only to vCloud targets of version 5.5 or newer.
`--targetPEM`				File path to a Privacy Enhanced Mail (.pem) file used to verify vSphere connections. Example: --targetPEM:<filename>.pem
`--targetSSLThumbprint`		N/A	vSphere	SSL thumbprint of the target. OVF Tool verifies the SSL thumbprint that it receives from the target, if this value is set.

Table 2-2. OVF Tool Command-Line Options (Continued)

Option's Long Name	Short Name	Source Types	Target Types	Description
--targetType	-tt	N/A	OVF, OVA, VMX, VI, vCloud, ISO, FLP, vApprun	Explicitly express that the target is OVF, OVA, VMX, VMX, vSphere, vCloud, ISO, FLP, or vApprun.
--vCloudTemplate				Create only a vAppTemplate.
--vService		OVF, OVA	N/A	Set a dependency on a vService provider in the OVF package, using the following syntax: --vService:<dependencyId>=<providerId>
--verifyOnly		All	N/A	Do not upload the source; just verify it. This only applies to vSphere 4.
--version	-v	N/A	N/A	Shows version information for OVF Tool. Binary option.
--viCpuResource		N/A	vSphere	Specify the CPU resource settings for VI locator targets. The syntax is --viCpuResource=<shares>:<reservation>:<limit>
--viMemoryResource		N/A	vSphere	Specify the memory resource settings for vSphere locator targets. The syntax is --viMemoryResource=<shares>:<reservation>:<limit>
--vmFolder	-vf	N/A	vSphere	The target virtual machine folder in vSphere inventory (for a datacenter).

Creating and Using the VM ID

When the parameters for one of the command line options includes the *VM ID*, this id refers to an attribute in the OVF descriptor file. Specifically, it is the id attribute of the VirtualSystem element that will appear in the OVF file that describes the VM you want to create or customize. If you are creating a VM, you need to specify the id in the descriptor file. For example, the id of the VM specified in the descriptor fragment below is *vm1*.

```
<ovf:VirtualSystem ovf:id="vm1">
   <ovf:Info>A virtual machine</ovf:Info>
   <ovf:Name>WinServer2012</ovf:Name>
   <ovf:OperatingSystemSection ovf:id="74" vmw:osType="windows8Server64Guest">
      <ovf:Info>Specifies the operating system installed</ovf:Info>
      <ovf:Description>Microsoft Windows Server 2012 (64-bit)</ovf:Description>
   </ovf:OperatingSystemSection>
   ........
```

For example, you need to use the VM ID when specifying the size of the memory for a VM.

```
> --memorySize:vm1=1024
```

If you are customizing an existing VM, look at the descriptor file to get the VM ID. You can also have the `ovftool` read an ovf file and extract the IDs before importing or deploying it.

```
>ovftool --verifyOnly --machineOutput <src ovf>
```

Specifying Disk ID to Set Size

When specifying disk sizes, you will need to specify the instance ID as well as the VM ID. The instance ID is the value of RASD InstanceID element of the virtual hardware section element describing the disk that should be resized.

```
<ovf:DiskSection>
   <ovf:Info>Virtual disk information</ovf:Info>
   <ovf:Disk ovf:capacity="4" ovf:capacityAllocationUnits="byte * 2^20"
      ovf:diskId="disk1" ovf:fileRef="disk1-file"
      ovf:format="http://www.vmware.com/interfaces/specifications/vmdk.html#streamOptimized"/>
</ovf:DiskSection>

<ovf:VirtualSystem ovf:id="vm1">
   .......
   <ovf:VirtualHardwareSection>
   <ovf:Info>Virtual hardware requirements</ovf:Info>
   <ovf:Item>
      <rasd:AddressOnParent>0</rasd:AddressOnParent>
      <rasd:Description>SCSI Hard disk</rasd:Description>
      <rasd:ElementName>SCSI Hard disk 1</rasd:ElementName>
      <rasd:HostResource>ovf:/disk/disk1</rasd:HostResource>
      <rasd:InstanceID>2000</rasd:InstanceID>
      <rasd:Parent>2</rasd:Parent>
      <rasd:ResourceType>17</rasd:ResourceType>
   </ovf:Item>
   .......
```

In the above example specifying instance ID "2000" (without quotes) would cause the disk with id "disk1" (without quotes) to be resized: --diskSize:vm1,2000=256 (set the size to 256).

Note that if multiple disk devices are backed by the same disk (i.e. the OVF contains multiple disk RASD items that refer to the same disk) you must specify the new size for all of these disk elements, not just one. Sharing disks between VMs is not common, but allowed in the OVF spec.

Note that you cannot shrink disks.

Specifying the Storage Profile ID

When you specify a storage profile, the ID refers to the UID of the storage profile in the vCD cell. This can be obtained using the vCD REST API.

More Help Topics

For more help, type: `--help <topic>`, where topics are:

locators	: detailed source and destination locator syntax
debug	: lists the debug settings
examples	: examples of usage
config	: syntax of configuration files
integration	: list of options primarily used when you execute the `ovftool` from another tool or shell script

Using the Log Settings

Use the OVF Tool's log options if you are not seeing the results you expect. The log options allow you to see the operations of the OVF Tool, and send the results to the console or to a file.

Two of the most commonly used options are: `--X:logFile` and `--X:logLevel`.

- Use the `--X:logFile=<filename>` option to log the complete `ovftool` session to a file
- Use the `--X:logLevel=<level>` option to control the verbosity of the logs

For example, you can use a command similar to this to write the log in a file called `ovftool-log.txt`:

```
>ovftool --X:logFile=ovftool-log.txt --X:logLevel=verbose LAMP.ovf
vi://localhost/Datacenter/host/Cluster
```

The following table lists all of the log options.

Log Option	Use option to:
--X:logFile=<*filename*>	Log internal events to a specified log file.
--X:logLevel=<*level*>	Log level. Valid values are: `none`, `quiet`, `panic`, `error`, `warning`, `info`, `verbose`, and `trivia`).
--X:logToConsole	Log internal events to console
--X:logTransferHeaderData	Add transfer header data to the log. Use with care. Default value is `false`

The OVF Tool includes 22 other debug options, that you can set to retrieve specific data. You can see all of the debug options and their definitions by running `ovftool --help debug`.

Specifying a Locator

A source or target locator points to a specific resource. Locators must specify a protocol, which defines how to reach the resource. Supported protocols are file access, vSphere, HTTP, HTTPS, and FTP.

File locators can point to an OVF package (`.ovf` or `.ova`), a virtual machine (`.vmx`). HTTP, HTTPS, or a vApprun workspace entity. FTP locators can point to OVF and OVA files. The resource type is determined from the filename suffix, unless one or both of the options `--sourceType` and `--targetType` are used explicitly.

vSphere locators can point to various resource types: virtual machines, vApps, hosts, clusters, or resource pools. For a source locator, the resource type must be a virtual machine or vApp. For a target locator, the resource type must be a host, cluster, or a resource pool. A vSphere locator is used for a vSphere server, vCenter Server, VMware Server, or an ESXi host.

At the command line, type **`--help locators`** to display the online help for locators.

Table 2-3 and Table 2-4 list the default extensions of the different source and target types, as well as which protocols are supported.

Table 2-3. Source Locator

Source Type	Default File Extension	Protocol	Example
OVF	`.ovf`	File, HTTP, HTTPS, FTP	/ovfs/my_vapp.ovf
OVA	`.ova`	File, HTTP, HTTPS, FTP	/ovfs/my_vapp.ova
VMX	`.vmx`	File	/vms/my_vm.vmx
vApprun	N/A	File	~/my_vApprun_workspace/MyVM
vCloud Director	N/A	HTTPS	vcloud://username:password@mycloud.org/ \ org=MyOrg&vdc=MyVDC&catalog=MyCatalog \ &vapp=myVapp
vSphere	N/A	vSphere	vi://username:pass@localhost/my_datacenter/vm/ \ my_vms_folder/my_vm_name

Table 2-4. Target Locator

Target Type	File Extension	Protocol	Example
OVF	`.ovf`	File	/ovfs/my_vapp.ovf
OVA	`.ova`	File	/ovfs/my_vapp.ova
VMX	`.vmx`	File (Source must be a single virtual machine)	/vms/my_vm.vmx
vApprun	N/A	File	~/my_vApprun_workspace/MyVM
vCloud Director	N/A	HTTPS	vcloud://username:password@mycloud.org/ \ org=MyOrg&vdc=MyVDC&catalog=MyCatalog \ &vapp=myVapp
vSphere	N/A	vSphere (If the vSphere target locator is on a VMware Server system, or directly on an ESXi host, the source must be a single virtual machine)	vi://username:pass@localhost/my_datacenter/vm/ \ my_vms_folder/my_vm_name

File Locators

File locators are the same for source and target. They are specified using ordinary path syntax.

Windows Path Syntax

On Windows, paths are specified as either absolute or relative.

This is an example of an absolute path on Windows:

```
C:\folder1\folder2\package.ovf
```

These examples show relative paths on Windows:

```
..\folder1\package1.ovf
package1.ovf
```

Linux and Mac OS Path Syntax

On Linux, paths are specified, similarly, as either absolute or relative.

The following is an example of an absolute path on Linux:

```
/folder1/folder2/package.ovf
```

The following are examples of relative paths on Linux:

```
../folder1/package1.ovf
package1.ovf
```

Using URIs as Locators

It is possible to specify file locations as a URI by prefixing the path with `file://`, as shown in the following examples:

```
file://c:\folder1\folder2\package.ovf (Absolute, Windows)
file:///folder1/folder2/package.ovf (Absolute, Linux)
file://package.ovf (Relative for both Windows and Linux)
```

Encoding Special Characters in URL Locators

When you use URIs as locators, you must escape special characters using % followed by their ASCII hex value. For instance, if you use a "@" in your password, it must be escaped with %40 as in vi://foo:b %40r@hostname, and a slash in a Windows domain name (\) can be specified as %5c.

HTTP, HTTPS, and FTP Locators

You can use HTTP, HTTPS, and FTP to refer to an OVF package (OVF or OVA file) on a Web server. You can only use these protocols to specify a source locator. In the following syntax, `protocol` is HTTP, HTTPS or FTP:

`protocol://username:password@host:port/<path to OVF package>`

It is possible to omit the user name and password from the locator. If needed, OVF Tool prompts you for them. If you use the standard port, it is not necessary to specify the port. Table 2-5 shows the standard ports.

Table 2-5. Standard Ports

Protocol	Port
HTTP	80
HTTPS	443
FTP	21

vSphere Locators

vSphere source locators point to a virtual machine or vApp within the virtual infrastructure. The vSphere target locator provides all required information for importing an OVF package or virtual machine into a cluster, host or resource pool. Both source and target locator use the same syntax:

`vi://<username>:<password>@<host>:<port>/<search-term>`

The server name and port can designate either a vCenter server, VirtualCenter server, VMware Server, or an ESXi host. If you omit credentials, in which case OVF Tool prompts you for them. Default installations of vCenter Server, VirtualCenter, and ESXi use port 443. If you are using the default port, you do not need to specify it. When using OVF Tool against a VMware Server, you must explicitly specify port 8333, which is the default port for VMware Server.

The search term has the following format:

```
<path>[?<query>=<value>]
```

If a query is not given, a VC inventory path lookup is performed using the specified path. Otherwise, the object matching the query is used. The meaning of the query depends on the object type. Table 2-6 shows the different values that you can use in the query field.

Table 2-6. Source and Target Values for All Query Types

Name	Query	Source	Target
BIOS	`bios`	BIOS ID of a virtual machine	BIOS ID of a host
Datastore	`ds`	Datastore path to a virtual machine	N/A
IP Address	`ip`	IP address of a virtual machine	IP address of a host
DNS	`dns`	DNS name of a virtual machine	DNS name of a host
Mo-Ref	`moref`	Managed object reference (vSphere specific identifier) of a virtual machine or vApp	Managed object reference (vSphere specific identifier) of a host, cluster, or resource pool

Table 2-7 shows example values for each query type.

Table 2-7. Examples of Query Values

Name	Query	Example Value
BIOS	`bios`	`vi://localhost?bios=234290984`
Datastore	`ds`	`vi://localhost/TestDatacenter?ds=[foo]/myvm/myvm.vmx`
IP Address	`ip`	`vi://localhost?ip=123.231.232.232`
DNS	`dns`	`vi://localhost?dns=production-vm3.example.com`
Mo-Ref	`moref`	`vi://localhost?moref=vim.VirtualMachine:vm-23423`

Note: For the same VM, the Managed Object Reference (MoRef) will be different for vCenter Server than it is for the ESXi host. For instance, the same VM can have a MoRef of 'vm-92' in vCenter Server and '118' in the ESXi host. Also note that the syntax contains 'vm-' for a vCenter Server MoRef. When the Motif is defined by vCenter Server, you can see it in the browser address and use it directly. The OVF Tool script doesn't define and parse the format of a MoRef.

You can enter a partial source locator if you do not know the entire inventory path. In this case, the tool fails but suggests possible inventory path completions.

Specifying the Inventory Path to a Virtual Machine or vApp

To specify an inventory path for a virtual machine or vApp, use the following syntax:

`<datacenter name>/vm/<folders>/<vm or vApp name>`

or

`<datacenter name>/host/<resource pool path>/<vm or vApp name>`

The use of the `vm` tag after the datacenter name specifies that you are locating a virtual machine or vApp in the VM and Template view. Use the host tag after the datacenter name if you are locating a virtual machine or vApp in the Host and Clusters view.

The following example shows an inventory path without any folders:

`MyDatacenter/vm/MyVM`

The following example shows an inventory path with two nested folders:

`MyDatacenter/vm/Folder 1/Sub Folder/MyVM`

Specifying the Inventory Path for a Cluster, Host, or Resource Pool

You can specify an inventory path for a host or a resource pool. You can nest resource pools similar to folders. To specify an inventory path for a host or a resource pool as part of target locators, use the following syntax:

`<datacenter name>/host/<host name>/Resources/<resource pool>`

- `host` and `Resources`. Fixed parts of the path.
- `Resources`. Specify only when a resource pool is specified.
- `<resource pool>`. Can take the value of one or more nested resource pools. If no resource pools are specified, the default resource pool for the host is used.

The following example is of an inventory path without a specified resource pool:

```
vi://username:pass@localhost/my_datacenter/host/esx01.example.com
```

The following example is of an inventory path with a specified resource pool:

```
vi://username:pass@localhost/my_datacenter/host/esx01.example.com/Resources/my_resou
rcepool
```

Note You must specify the /host/ section of an inventory path when using a vi destination locator. If you are specifying the destination of a resource pool, you must include the /Resources/ section of the path.

vCloud Director Locators

The syntax for vCloud locators are the same as for other locators:

```
vcloud://username:password@host:port?org=name_of_org&vapp=name_of_deployed_vapp&
catalog=name_of_catalog&vappTemplate=name_of_vapp_template_in_catalog&vdc=name_of_vdc
```

Some of the options are not needed if there is only one virtual datacenter to choose from. If there are more than on datacenter, the catalog option is required. The `org` option is mandatory, because it is used to log in to vCloud Director.

Note OVF Tool supports all source types for vCloud Director 1.5. For vCloud Director 1.0, OVF Tool only supports OVF/OVA/vCloud sources. OVF Tool does not support vi, vmx, or vapprun sources for vCloud Director 1.0.

Examples of vCloud Locators

The following example uploads and deploys an OVF named test into vCloud Director and names the vApp my_test1.

```
ovftool /tmp/test.ovf vcloud://user1:password@10.99.99.99:7443?org=o&vapp=my_test1
```

This example exports a vCloud Director vApp to the OVF file /tmp/test1.ovf

```
ovftool vcloud://user1:password@10.10.99.99.99:7443?
org=o&vapp=my_test1 /tmp/test1.ovf
```

If you use a network, you map the network in the usual way:

```
--net:sourceNET=targetNET
```

You also apply properties in the usual way.

Partial Locators

When using OVF Tool, it is often not necessary to specify source and target types as long as certain filename conventions are used. It is possible to the ignore locator type and specify the source and target explicitly using the arguments **--sourceType=**... and **--targetType=**.

OVF Tool assumes the locator type based on the following rules:

- If the name starts with `vcloud://`, OVF Tool assumes vCloud Director type.
- If the name starts with `vi://`, OVF Tool assumes vSphere type.
- If the name ends with `.ovf`, OVF Tool assumes OVF type.
- If the name ends with `.vmx`, OVF Tool assumes VMX type.
- If the name ends with `.ova`, the OVF tool assumes OVA type.
- If the locator is a file path to a directory that represents a vApprun workspace or an entity in a vApprun workspace, then OVF Tool assumes vApprun type.

Similarly, source and target types can be inferred from folder locators. OVF Tool assumes the type using the following rules:

- If the source locator is a folder, OVF Tool assumes that the source is an OVF package and that the OVF descriptor is called the same as the folder, for example, `my-ovf/my-ovf.ovf`.
- If the source is an OVF package and the target locator is a directory, such as `MyVirtualMachines/`, OVF Tool assumes that the target is a VMX locator. The created VMX/VMDK file is put in a directory with the target name, for example, `MyVirtualMachines/MyVM/MyVM.vmx`.
- If the source is a VMX locator and the target locator is a directory, OVF Tool assumes that the target is an OVF package.
- If the source is a vSphere locator, and the target locator is a directory, OVF Tool assumes that the target is an OVF package.

OVF Tool supports partial vSphere locators when deploying or exporting. For an incomplete locator path, the tool suggests completions at the command line. Example: Partial vSphere Locators at the Command Line shows the command-line dialog when partial locators are used.

Example: Partial vSphere Locators at the Command Line

```
> ovftool LAMP.ovf vi://localhost/
Opening source: LAMP.ovf
Opening target: vi://user@localhost/
Error: Found wrong kind of object (Folder)
Possible completions are:
  Datacenter/
  Remote Datacenter/
  Secondary Datacenter/

> ovftool LAMP.ovf vi://localhost/Datacenter
Opening source: LAMP.ovf
Opening target: vi://user@localhost/Datacenter
Error: Found wrong kind of object (Datacenter)
Possible completions are:
  vm/
  host/

> ovftool LAMP.ovf vi://localhost/Datacenter/host
```

```
Opening source: LAMP.ovf
Opening target: vi://user@localhost/Datacenter/host
Error: Found wrong kind of object (Folder)
Possible completions are:
  host1.foo.com/
  host2.foo.com/

> ovftool LAMP.ovf vi://localhost/Datacenter/vm/host1.foo.com
```

OVF Tool supports partial vSphere locators when deploying or exporting. For an incomplete locator path, the tool suggests completions at the command line. Example: Partial vCloud DirectorLocators at the Command Line shows the command-line dialog when partial locators are used. First, OVF Tool signals that there is more than one virtual datacenter present, then multiple catalogs, then multiple networks. At each attempt, you must select one of the options that OVF Tool presents.

Example: Partial vCloud DirectorLocators at the Command Line

```
ovftool LAMP.ovf vcloud://jd:PASSWORD@example.com:443/?org=myOrg&vapp=test1
Opening OVF source: LAMP.ovf
Warning: No manifest
file

Opening vCloud target: vcloud://js:PASSWORD@example.com:
443/

Error: Multiple VDCs found. Possible VDC completions
are:


orgVdc



orgVdc2


Completed with errors

ovftool LAMP.ovf "vcloud://jd:PASSWORD@example.com:443/?org=myOrg&vapp=test1&vdc=orgVdc"
Opening OVF source: LAMP.ovf
Warning: No manifest
file

Opening vCloud target: vcloud://js:PASSWORD@example.com:443/
Error: Multiple catalogs found. Possible catalog completions
are:


catalog



catalog2
```

```
Completed with errors

"vcloud://jd:PASSWORD@example.com:443/?org=myOrg&vapp=test1&vdc=orgVdc&catalog=catalog"
Opening OVF source: LAMP.ovf
Warning: No manifest
file

Opening vCloud target: vcloud://js:PASSWORD@example.com:443/
Error: Multiple networks found on target. Possible completions
are:


extNet2



extOrgNet



intNet2


 intnet
Completed with errors

ovftool –-net:"VM Network=intnet" LAMP.ovf "vcloud://jd:PASSWORD@example.com:443/
?org=myOrg&vapp=test1&vdc=orgVdc&catalog=catalog"
Opening OVF source: LAMP.ovf
Warning: No manifest
file

Opening vCloud target: vcloud://js:PASSWORD@example.com:443/
Deploying to vCloud: vcloud://js:PASSWORD@example.com:443/
Disk Transfer Complete
Completed successfully
```

Configuration Files

OVF Tool has many options. Rather than repeatedly entering long commands on the command line, you can create a configuration file. A configuration file uses the following syntax:

```
option1=value
...
#comment
optionN=value
```

The following is an example of a configuration file:

```
proxy=http://proxy.example.com
datastore=storage-test42
# Comment on something
locale=dk
```

You can create local or global configuration files. A local configuration file has the `.ovftool` suffix and is read in the folder from which you invoke OVF Tool. A global configuration file is per user.

On pre-Vista Windows, the global configuration file is read from the following location:

`C:\Documents and Settings\$USERNAME\VMware\ovftool.cfg`

On post-Vista Windows, the global configuration file is read from the following location:

`C:\Users\$USERNAME\AppData\Roaming\VMware\ovftool.cfg`

On Linux and Mac, the global configuration file is read from the following location:

`$HOME/.ovftool (example: ~/.ovftool)`

When using configuration files, globally defined options are overwritten by locally defined and command-line options. Locally defined options are overwritten by command-line options. Note: there are some variables, such as proxy, that do not allow you to specify the options in multiple places. If the ovftool stops running with the error, "already `exists`", it is very likely that you have specified options on the command line and in the configuration file.

You can use the **`ovftool --help config`** command to get information about how to use a configuration file. In addition, the current contents of the global configuration file as well as any local configuration file is shown.

Handling Authentication

OVF Tool generates AUTHENTICATION output messages if access to a resource requires a username or password. For example, a proxy server, a vSphere or vCloud locator, or an authenticated URL require usernames and passwords. OVF Tool only generates AUTHENTICATION messages for resources where passwords are not explicitly provided as part of the locator or as command-line arguments.

OVF Tool can authenticate the following types of objects:

- source locators
- target locators
- proxyServer

For source and target locators, you must provide the username on the command-line. If you do not provide a password, OVF Tool generates an AUTHENTICATION message and you must provide the password on STDIN. If the proxy server requires authentication, you must provide both the username and password on STDIN

OVF Tool supports the following commands on STDIN:

```
For the source password:

PASSWORDSOURCE
password

For the target password:

PASSWORDTARGET
password

For the proxy:

PASSWORDPROXY
username password
```

For an example of the output of running `machineOutput` in authentication mode, see Output from Running machineOutput in Import Mode.

Launching the OVF Tool as a Helper Process

You can use the integration options to make it more convenient to launch OVF Tool as a helper process to a client of the vSphere Web Services API, such as a script using Perl bindings.

If you use the `--I:morefArgs` argument, the values for `--vmFolder`, `--network`, `--net`, and `--datastore` are interpreted as MoRefs instead of names, as shown in the following example:

```
> ovftool --name=vm5 \
                --I:morefArgs \
                --net:VM Network=vim.Network:network-12 \
                --datastore=vim.Datastore:datastore-17 \
                c:\temp\vm1\ \
                vi://root:@localhost?moref=vim.ResourcePool:resgroup-42
```

Use the `--I:sourceSessionTicket` or `--I:targetSessionTicket` options to authenticate with a session ticket retrieved from `SessionManager.AcquireSessionTicket`, when using the vSphere source or destination.

Examples of OVF Tool Syntax

3

This chapter provides many examples of OVF Tool usage, that are divided into the following categories:

You can see similar examples within the OVF Tool, by typing **`--help examples`** on the command line while you are in the directory where the `ovftool` script is running.

This chapter includes the following topics:

Supported File Types and Package Formats

As discussed in Chapter 1, you can use the OVF Tool to package virtual machines as vApps (ready-to-use virtual machines with operating systems and/or applications). The package formats supported by the OVF Tool can be read and/or imported by other VMware and third-party software.

The table below describes each of the supported formats:

Table 3-1. Supported File and Package Types for OVF Tool Input and Output

Package Type	Full Name	Usage
OVF (.ovf)	Open Virtualization Format	National ANSI standard for packaging software for virtual machines, originally created by an industry task force known as the Distributed Management Task Force (DTMF). An OVF package includes: a descriptor file, optional manifest and certificate files, optional disk images, and optional resource files (such as ISOs). The disk image files can be files in VMware's `.vmdk` disk image format or in any other supported disk image format. OVF packages can be used by the software of any hypervisor or processor architecture that supports this format.
OVA (.ova)	Open Virtual Appliance	A TAR archive that contains an OVF package.
VMX (.vmx)	Virtual Machine Configuration File	When you create a new virtual machine, this file is created to store information about the operating system, disk sizes, networking, and virtual hardware. Files in this format and the `.vmdk` format are sometimes referred to together as, 'VMware runtime format'.
VMDK (.vmdk)	Virtual Machine Disk	Files with this extension may contain disk characteristics (,vmdk), contents (-flat.vmdk), or snapshot files (-delta.vmdk). These files are called out on the OVF Tool command line, but may exist within the package.
VI (vi://)	VMware Infrastructure	This is an older term that originated with ESX 3, but is still seen in the command line syntax for the OVF Tool. As an OVF command line option, 'vi//' is used before the credentials and path to a server.
vCloud	vCloud Director format	The vCloud Director REST API makes basic transfer between clouds possible using OVF packages, which preserve application properties, networking configuration and other settings.
ISO (.iso)	Optical Image File	An ISO archive is a CD/DVD image. Creating a package as an ISO image allows you to install a virtual appliance using a CD ROM drive. This type of archive is called an ISO because it was created by the International Standards Organization's 9660 standard.
FLP (.flp)	Floppy Disk Image File	Use this format if you need to transfer data from a floppy drive or to the virtual machine floppy drive. Instructions are available in Knowledge Base article 1739.
vApprun	vApprun	This format allows you to run a virtual appliance on VMware Workstation or Fusion. You can use the OVF Tool to convert vApps to the vApprun format, and you can use VMware Workstation to convert vApps to an OVF format.

Use the OVF Tool with the Target Type option to specify the target out as OVF, OVA, VMX, VI, vCloud, ISO, FLP, vApprun.

In this following example, the target type is set to the 'vmx' or VMware runtime format (.vmx and .vmdk files)

```
> ovftool –tt=vmx /ovfs/my_vapp.ovf /vms/
```

The resulting files are: `/vms/my_vapp/my_vapp.vmx` and `/vms/my_vapp/my_vapp.vmdk files (like the contents of a typical virtual machine directory).`

Changing File or Package Formats

The examples below show you how to convert from an existing format to a different format.

Converting an OVF Package to an OVA Archive

To convert an OVF package, to a single OVA archive, use the following syntax:

```
> ovftool /ovfs/my_vapp.ovf /ovfs/my_vapp.ova
```

Converting an OVA Archive to OVF Package

To extract an OVA archive to an OVF package, use the following syntax:

```
> ovftool /ovfs/my_vapp.ova /ovfs/my_vapp.ovf
```

Converting VMX Format to an OVF Package

To convert a virtual machine in VMware runtime format (.vmx and .vmdk files) to an OVF package, use the following syntax:

```
> ovftool /vms/my_vm.vmx /ovfs/my_vapp.ovf
```

The result is located in `/ovfs/my_vapp.ovf`

Converting VMX Format to an OVA Archive

To convert VMX files to an OVA archive, use the following syntax:

```
> ovftool vmxs/Nostalgia.vmx ovfs/Nostalgia.ova
```

Converting an OVA Archive to VMX Format

To convert an OVA archive to VMware runtime format (.vmx and .vmdk files), use the following syntax:

```
> ovftool https://my_ovf_server/ovfs/my_vapp.ova /vm/my_vm.vmx
```

Converting an OVF Package to VMX Format

To convert an OVF package to VMware runtime format (.vmx and .vmdk files), use the following syntax:

```
> ovftool http://www.mycompany.com/ovflib/BigDemo.ovf x:/myvms/BigDemo.vmx
```

Because the source is an OVF package, you can specify it as a URL or a local file path.

If you convert an OVF package to a VMX format without specifying the target directory, OVF Tool creates a directory using the OVF package name and writes the `.vmx` and `.vmdk` files in it.

```
> ovftool "Windows 7.ovf" .
```

The VMX file is written at `Windows 7/Windows 7.vmx`.

You can also convert from an ovf format to a vmx format using a URL, as shown:

```
> ovftool https://my_ovf_server/ovfs/my_vapp.ova /vm/my_vm.vmx
```

Installing an ESXi host from an OVF Package

To install an OVF package as an ESXi host, use the following syntax:

```
> ovftool /ovfs/my_vapp.ovf vi://username:pass@my_esx_host
```

(Uses default mappings.)

Installing an ESXi host from an OVF Package on a Web Server

To install an OVF package on a web server as an ESXi host, use the following syntax:

```
> ovftool http://my_ovf_server/ovfs/my_vapp.ovf vi://username:pass@my_esx_host
```

(Uses default mappings.)

Installing an ESXi host or Adding Files from a VMX Format

To install an ESXi host from a VMware runtime format (.vmx and .vmdk files), or to add .vmx files to an ESXi host, use the following syntax (uses default mappings):

```
> ovftool /ovfs/my_vm.vmx vi://username:pass@my_esx_host
```

```
> ovftool Nostaliga.vmx vi://user:pwd@host/Datacenter/host/host1.foo.com
```

Installing a vCenter Server or Adding Files from an OVF Package

To install or add files to a vCenter Server from an OVF package, use the following syntax (uses a managed ESXi host's IP address):

```
> ovftool /ovfs/my_vapp.ovf vi://username:pass@my_vc_server/?ip=10.20.30.40
```

Converting a VM on ESXi or vCenter Server to an OVF Package

This example uses a datastore location query to convert a VM (located on a vCenter Server) to OVF format.

```
> ovftool vi://username:pass@my_vc_server/my_datacenter?ds=[Storage1] foo/foo.vmx
c:\ovfs\
```

or

```
> ovftool
vi://username:pass@my_host/my_datacenter/vm/my_vm_folder/my_vm_name /ovfs/my_vapp.ov
f
```

Installing vCenter Server from an OVF Package Using an Inventory Path

This example uses a vSphere inventory path to install or add files to a vCenter Server from an OVF package.

```
> ovftool /ovfs/my_vapp.ovf
vi://username:pass@my_vc_server/my_datacenter/host/my_host
```

Setting OVF Package Properties

The following sections show you how to set properties for OVF packages.

Setting OVF Properties When Deploying to vSphere or vCloud Director

OVF descriptors can contain configuration properties for the deployed OVF package. You can set only one property at a time, but you can have multiple instances of the option per command. The property option has the following syntax:

```
--prop:<option>=<value>
```

This example sets two properties: the administrator's email address and the number of concurrent sessions.

```
> ovftool --prop:adminEmail=john@example.com --prop:concurrentSessions=200 package.ovf vi://localhost/?
dns=fast-esx=host1.example.com
```

Setting OVF Network Maps When Deploying to vSphere

OVF descriptors can use symbolic identifiers for network names. These identifiers must be mapped to a network that is available on the chosen vSphere platform. If only one network is available on the target and only one network is described in the OVF descriptor, OVF Tool selects that network automatically. In this case, you do not need to specify a network map. The `--net` option has the following syntax:

```
--net:<OVF network name>=<target network name>
```

In the following example, a network is selected.

```
> ovftool --net:"Example net 1"="VM Network" <source> <vSphere locator>
```

If the OVF descriptor only specifies one network name, you can specify the target network name of the network mapping, as in the following example:

```
> ovftool --network="VM Network" <source> <vSphere locator>
```

Setting a vService Dependency

You can dependency so that your application deploys as a service using the following option:

```
ovftool --vService:vDep1=provider_1 /ovfs/my_vapp.ovf
     vi://username:pass@localhost/my_datacenter/host/esx01.example.com
```

Modifying an OVF Package

The following sections show you how to modify OVF packages.

Renaming the OVF Package

You can rename an OVF package by converting the OVF to an OVF. This action also renames all the disk names and changes the references in the OVF descriptor.

```
> ovftool "Windows 7.ovf" win7.ovf
```

Omitting Disks in the VMware OVF Tool Output

If you want only information about the OVF descriptor and not about the disks that it refers to, you can suppress output. The following example command omits disk output and simply copies the OVF descriptor and any message bundle files that might be associated with it:

```
> ovftool --noDisks http://example.com/ovf/InterestingVirtualAppliance package.ovf
```

Compressing an OVF Package

For maximum compression of an OVF package with multiple virtual machines, set both the `--compress=9` and `--makeDeltaDisks` options. The following are examples of using maximum compression:

```
> ovftool --compress=9 --makeDeltaDisks package.ovf output-dir
> ovftool --compress=9 --makeDeltaDisks vi://localhost/dc/vm/VirtualAppDemo output-dir/
```

If the source contains only a single virtual machine, the `--makeDeltaDisks` option does not yield any compression boost. In this case, the `--compress=9` option gives maximum compression.

Chunking or Splitting OVF Packages

Some file systems have a restriction on maximum file size. For example, FAT32 allows files only up to 2GB.You can split the OVF files from a generated package into pieces of a specified maximum size. The default measurement is megabytes (keyword `mb`). You can specify other units using one of the following keywords:

Unit	Keyword
Bytes	b
Kilobytes	kb
Gigabytes	gb

For example, to create an OVF package optimized for a FAT32 file system, use the following command:

```
> ovftool ––chunkSize=2gb <source> package.ovf
```

Each file chunk has a sequentially numbered suffix. For example, for a 6GB disk, the chunks have these names:

```
disk1.vmdk.000000000, disk1.vmdk.000000001, disk1.vmdk.000000002
```

Deploying OVF Packages

The following sections show you how to deploy OVF packages.

Deploying an OVF Package Directly on an ESXi Host

The following command deploys an OVF package on an ESXi host.

```
> ovftool package.ovf vi://my.esx–machine.example.com/
```

If your host has multiple data stores, use the -ds option:

```
> ovftool package.ovf –ds=storage1 vi://my.esx–machine.example.com/
```

See also Run OVF Tool from ESXi instead of vCenter.

Deploying an OVF Package to vCenter Server

The following command deploys testVM.ovf from a local Windows disk to a data store named storage1 in host (12.98.76.54) from vCenter (12.34.56.789). the VM will be named `myVM` on the host. (To test this on your system, replace the incorrect IP addresses in this example with your actual data store and host IP addresses.)

```
> ovftool –ds=storage1 –n=myVM C:\testVM.ovf vi://user1:passwd@12.34.56.789/?
ip=12.98.76.54.
```

Deploying an OVF Package and Powering It On

OVF Tool can power on a virtual machine or vApp after deployment. This action can be done on all supported platforms. The following example powers on the VM or vApp on a particular host through vCenter Server:

```
> ovftool --powerOn package.ovf vi://MyvCenterServer/?dns=fast-esx-
host1.example.com.
```

Deploying an OVF Package into vCloud Director

You can deploy an OVF package from OVF Tool into vCloud Director. The following example connects to vCloud Director and deploys the OVF package LAMP.ovf.

```
> ovftool --net:"VM Network=intnet" LAMP.ovf "vcloud://jd:PASSWORD@example.com:443/?
org=myOrg&vapp=test1&vdc=orgVdc&catalog=catalog"
```

Deploying an OVF Package into a vApprun Workspace

A vApprun workspace allows Workstation and Fusion users to run vApps. It provides a complete vApprun execution environment, that includes nested vApps, OVF properties, and an OVF environment. The environment is fully compatible with vSphere 4 and all later releases.

Read more about vApprun at: http://labs.vmware.com/flings/vApprun.

To deploy an OVF package into a vApprun workspace, simply use a target locator that points to your vApprun workspace, as shown in the following example:

```
> ovftool myOvfPackage c:\My_vApprun_workspace\
```

A common scenario is that the current directory is the vApprun workspace (since all vApprun commands are relative to this), so you can just use a "." as the target locator, as shown in the following example:

```
> ovftool http://www.mycompany.com/ovflib/BigDemo.ovf.
```

Importing an OVF Package

The following sections show you how to import OVF packages.

Importing an OVF File into a vCloud instance

This section provides two examples of importing using a URL to create a vCloud instance from an OVF file. The first example names the resulting vApp as 'vApp'.

```
> ovftool http://my_ovflib/vm/my_vapp.ovf \
           vcloud://username:pass@my_cloud?org=MyOrg&vdc=MyVDC&catalog=MyCatalog&vapp=myVapp
```

(Imports an OVF from http into a vCloud instance and names the vApp myVapp)

The second example also creates a vApp template:

```
> ovftool http://my_ovflib/vm/my_vapp.ovf \
vcloud://username:pass@my_cloud?org=MyOrg&vdc=MyVDC&catalog=MyCatalog&vappTemplate=myTemplate
```

(This imports an OVF from http into a vCloud instance and creates a vApp template)

Importing a Virtual Machine from vSphere to vCloud

The following command imports a virtual machine from vSphere into a vCloud instance and names the resulting vApp 'myVapp'.

```
> ovftool vi://username:pass@my_host/my_datacenter/vm/my_vm_folder/my_vm_name \
           vcloud://username:pass@my_cloud?org=MyOrg&vdc=MyVDC&catalog=MyCatalog&vapp=myVapp
```

Importing VMX Files into a vApprun Workspace

To import a VMX format into a vApprun workspace, use the following syntax:

```
> ovftool /virtualmachines/MyVM.vmx ~my_vApprun_workspace/
```

Exporting Virtual Machines to OVF Packages

The following sections show you how to export virtual machines within OVF packages.

Exporting a Virtual Machine from a vCloud instance to an OVF Package

The following command exports a Virtual Machine from a vCloud instance into an OVF package.

```
> ovftool vcloud://username:pass@my_cloud?org=MyOrg&vdc=MyVDC&catalog=MyCatalog&vapp=myVapp \
           /ovfs/myVapp.ovf
```

Exporting a Running Virtual Machine or vApp from vSphere

You must power off a virtual machine or vApp before exporting it. The following example locates the virtual machine or vApp based on its DNS name through the vCenter Server and powers it off:

```
> ovftool --powerOffSource vi://MyvCenterServer/?dns=test-vm test-vm.ova
```

Note This option does not perform a shutdown of the operating system. This is only a power off operation.

Exporting a vApprun Entity to an OVF Package

Both virtual machine and vApp entities in your vApprun workspace can be exported as OVF packages, as shown in the following example:

```
> ovftool c:\My_vApprun_workspace\BigDemo c:\ovflib\
```

Prepend the name of the entity to export to the path. If the current directory is the vApprun workspace, you only specify the name, as shown in the following example. This example takes advantage of the fact that any source locator can be used with any destination locator. Thus, the vApp transfers directly from the vApprun workspace to the vCenter installation.

```
> ovftool BigDemo vi://MyvCenterServer/...
```

Note vApprun does not keep the same level of meta-data around as vSphere. Thus, the vApprun-created OVF packages will not contain any EULAs, description of properties, and such.

Displaying Summary Information

To display a summary of information about the OVF package [in probe mode], use the following syntax:

```
> ovftool https://my_ovflib/vm/my_vapp.ovf
```

(shows summary information about the OVF package [probe mode])

Validating an OVF 1.0 or OVF 1.1 Descriptor

If you are generating OVF 1.0 or OVF 1.1 descriptors manually, you can check whether the descriptors comply with OVF 1.0 or OVF 1.1. The following examples show how to validate descriptors:

```
> ovftool --schemaValidate package.ovf
> ovftool --schemaValidate package.ova
> ovftool --schemaValidate http://example.com/folder1/package.ovf
> ovftool --schemaValidate http://example.com/folder1/package.ova
```

If everything is correct, OVF Tool shows the result of probing OVF. Otherwise, it shows warnings and errors.

Important Being compliant with OVF 1.0 or 1.1 is only part of the requirements for a valid OVF package. Schema validation does not check for all the requirements specified in the OVF 1.0 and OVF 1.1 specifications.

Downloading an OVF Package from a Protected Web Site

The OVF Tool can read sources given by a URL using both HTTP and HTTPS. You access it with the user name and password. The following example downloads the LAMP OVF package and puts it in an OVA package. If you omit the user name and password, the OVF Tool will prompt you for them.

```
> ovftool https://user:pass@example.com/repository/ovf/LAMP.ovf LAMP.ova
```

Using a Proxy

You can specify a proxy for the OVF Tool. The following examples show the use of the `--proxy` option:

```
> ovftool --proxy=proxy.example.com http://external-site.com/ovf/package.ovf
> ovftool --proxy=http://proxy.example.com http://external-site.com/ovf/package.ovf
```

The OVF Tool allows proxies that require authentication. Credentials are supplied in the proxy path as shown in the following example:

```
> ovftool --proxy=user:pass@proxy.example.com http://external-
site.com/ovf/package.ovf
```

You can omit the username and password for a proxy server that requires authentication. OVF Tool prompts for them. If you are using vSphere or vCloud as the locator for a source or target, you have to add the UseProxy option: --X:viUseProxy for vSphere or --X:vCloudUseProxy for vCloud.

```
> ovftool --X:viUseProxy --proxy=proxy.example.com package.ovf vi://my.esx-machine.example.com/
> ovftool --X:vCloudUseProxy --proxy=proxy.example.com package.ovf
"vcloud://USERNAME:PASSWORD@example.com:443/?org=myOrg&vapp=test&vdc=orgVdc&catalog=catalog"
```

Overwriting a Running Virtual Machine or vApp from vSphere

The OVF Tool supports overwriting existing targets. If a target virtual machine or vApp has the same name as the source, OVF Tool overwrites the target when the `--overwrite` option is specified. If the target virtual machine or vApp is running, OVF Tool cannot overwrite it. OVF Tool does not automatically power off the target. To power off the target before overwriting it, use the `--powerOffTarget` option.

```
> ovftool --overwrite --powerOffTarget package.ovf vi://localhost/?dns=production-
host.example.com
```

You can also power on the newly written virtual machine or vApp at the same time. In the following example, the target machine is powered off and deleted, the package.ovf is imported, and the imported virtual machine or vApp is powered on.

```
> ovftool --overwrite --powerOffTarget --powerOn package.ovf vi://localhost/?
dns=production-host.example.com
```

Canceling the VMware OVF Tool While it Is Running

To cancel OVF Tool while it is running, type **`Ctrl-C`**. This halts OVF Tool and cleans up any generated files.

OVF Package Signing

4

A valid OVF signature requires two special files, a manifest (`.mf`) file that contains the SHA hash codes of all the files in the package (except the `.mf` and `.cert` files), and a certificate file (`.cert`) that contains the signed SHA of the manifest file and the X.509 encoded certificate. This appendix specifies how to use OpenSSL and VMware OVF Tools commands to sign and validate OVF packages.

This chapter includes the following topics:

Creating an RSA Public/Private Key Pair and Certificate

To sign a package, a public/private key pair and certificate that wraps the public key is required. The private key and the certificate, which includes the public key, is stored in a `.pem` file.

The following OpenSSL command creates a `.pem` file:

```
> openssl req –x509 –nodes –sha256 –days 365 –newkey rsa:1024 –keyout myself.pem –out myself.pem
```

Note No password is necessary. To include a password, remove the ––`nodes` option.

Example: Myself.pem File Contents shows the contents of the `myself.pem` file.

Example: Myself.pem File Contents

```
-----BEGIN RSA PRIVATE KEY-----
    MIICXAIBAAKBgQDe0dCCKNfQ45+D0ezGGAuVSbhE8buqFCQnQnfi27Wt6bu4DhcE
    bQtjgfzuEpcl4e31txJcu18XTv4icRL74DP7i2pMN2UVj6DZW/B7jIw4UPG2g96f
    ...
    -----END RSA PRIVATE KEY-----
    -----BEGIN CERTIFICATE-----
    MIIC5DCCAk2gAwIBAgIJAKgUiZPOajC0MA0GCSqGSIb3DQEBBAUAMFYxCzAJBgNV
    BAYTAkRLMRMwEQYDVQQIEwpTb21lLVN0YXRlMQ8wDQYDVQQHEwZBYXJodXMxITAf
    ...
    -----END CERTIFICATE-----
```

To display the contents of a `.pem` file at the command line, type the following:

`>openssl x509 –text –noout –in <filename>.pem`

The contents of the file display as follows:

```
Certificate:
     Data:
        Version: 3 (0x2)
        Serial Number:
....
```

To create a trusted certificate, use the OpenSSL command, omitting the `––x509` option. This creates a certificate request in a `.pem` file that you can send to any public authority, such as Verisign.

Signing an OVF Package

Signing an OVF package enables the person deploying it to validate the authenticity of the OVF package. Once the package is signed, OVF package files cannot be changed, without invalidating the signature. When a package comes from a trusted source and has a valid OVF signature, you can deploy the package knowing it has not been tampered with.

Signing an OVF package requires a `.pem` file that contains a private key and a certificate, as shown in section Creating an RSA Public/Private Key Pair and Certificate.

To sign a generated OVF package, include the --privateKey option. The option syntax is shown in the following example:

```
> ovftool ––privateKey=<path to .pem file> <source> <output OVF or OVA file>
```

When this option is used, OVF Tool uses the private key and certificate to generate a signature based on the SHA digest of each file that is included in the OVF package, including the OVF descriptor itself.

OVF Tool generates an additional `.cert` file with a signed SHA signature and the certificate used to sign it. Example: Certificate File Created by OVF Tool shows an example of the `.cert` file generated by OVF Tool.

Example: Certificate File Created by OVF Tool

```
SHA256(signed–
package.mf)=5d9a307f0acdc1a424079eb38ff8954c153f978e599ed374dd784c853bab1856415fa16ef378bde3487cd5dfa4d
11a3017eda91886f98e3bba3adc2f4e28ce6d0ba3a19eef80ac0729511311603dcb221f9ba7a6008f1a87fe15ebf3699c8a8744
bd05c43b1387dd53d73723e7f0a3720d489e147e31c4570d15fb7a3beae770
–––––BEGIN CERTIFICATE–––––
MIIDTzCCArigAwIBAgIJAKDgFLg9WvBwMA0GCSqGSIb3DQEBBQUAMHkxCzAJBgNV
BAYTAkRLMQ8wDQYDVQQHEwZBYXJodXMxFTATBgNVBAoTDFZNd2FyZSwgSW5jLjEM
MAoGA1UECxMDVklNMREwDwYDVQQDEwhLcmlzdGlhbjEhMB8GCSqGSIb3DQEJARYS
a2xhc3NlbkB2bXdhcmUuY29tMB4XDTA5MDMwNjEzMDUwNFoXDTEwMDMwNjEzMDUw
NFoweTELMAkGA1UEBhMCREsxDzANBgNVBAcTBkFhcmh1czEVMBMGA1UEChMMVk13
YXJlLCBJbmMuMQwwCgYDVQQLEwNWSU0xETAPBgNVBAMTCEtyaXN0aWFuMSEwHwYJ
KoZIhvcNAQkBFhJrbGFzc2VuQHZtd2FyZS5jb20wgZ8wDQYJKoZIhvcNAQEBBQAD
```

```
gY0AMIGJAoGBAM2xxX9a1YITiiRrxpXGg9xbEP4Oepcs71ZcNp8Z3mQIb95mpEc6
SZemmjOsqwpkvV/82RALOBgmJ/hot1noSkiAZi0liPmX1M0BU3OS/pSim7VNKBmV
SUJfOC4T6/MygVpyfkSUhB5EWx0JCUvowRex6Ytl220MOGcXnLpvdfO9AgMBAAGj
gd4wgdswHQYDVR0OBBYEFM2KkX7pWTQmMg+iD6HWMOZRLrfJMIGrBgNVHSMEgaMw
gaCAFM2KkX7pWTQmMg+iD6HWMOZRLrfJoX2kezB5MQswCQYDVQQGEwJESzEPMA0G
A1UEBxMGQWFyaHVzMRUwEwYDVQQKEwxWTXdhcmUsIEluYy4xDDAKBgNVBAsTA1ZJ
TTERMA8GA1UEAxMIS3Jpc3RpYW4xITAfBgkqhkiG9w0BCQEWEmtsYXNzZW5Adm13
YXJlLmNvbYIJAKDgFLg9WvBwMAwGA1UdEwQFMAMBAf8wDQYJKoZIhvcNAQEFBQAD
gYEANaNxv4QrN7iI0rDCordYDh1G7Z3jl28ntSoxehGmz6ghYAfBNhTVhWUZuX9X
UXKn8QltOF/Ynijuo6JTJwO/5V1o6TAaCmFahDW/Om02AXPdSbw4UQdidGmmgrAs
DYVQz2CNPk2YbkXITNeGBNHomTqsVU7MGDjReu96+V6O2zY=
-----END CERTIFICATE-----
```

Validating an OVF Package

If an OVF certificate file is present, OVF Tool always verifies if the signature fits the SHA digest of the files in the package and tests the authenticity of the certificate.

To quickly validate the authenticity of an OVF package, use the probe mode as shown in the following example:

```
> ovftool signed-package.ovf
```

Using the VMware OVF Tool Probe Mode

5

This chapter includes the following topics:

About OVF Tool Probe Mode

Probe mode reveals information about the content of a source. You can probe OVA and OVF packages, VMX, and vSphere source types. You can use the information gathered to find out how it can be configured when you deploy it.

To use the probe feature, omit the target locator when invoking OVF Tool. For example, at the command line, type: **`ovftool LAMP.ovf`**. The tool displays all available information about the `LAMP.ovf`.

When probe mode is used on an OVF or OVA package, OVF Tool also validates the certificate file, if present.

As part of the information displayed in probe mode, the EULA is displayed by default. To prevent the EULA from displaying, use the `--hideEula` option.

```
> ovftool --hideEula LAMP.ovf
```

Example of Probe Mode

The following example shows the result of probing the `LAMP.ovf` package.

```
OVF version:   1.0
VirtualApp:    true
Name:          LAMP running PHP-Fusion
Version:       0.1
Vendor:        VMware Aarhus
Product URL:   http://example.com/ovf/1.0/LAMP/readme.txt

Annotation:  This vApp offers the programming environment stack: Linux, Apache,
           MySQL and PHP programming environment -- LAMP. More specifically
           the vApp contains a Database server running MySQL and Web server
           VM running Apache2 and PHP.

End-user License Agreements:
```

```
  EULA for LAMP.

Download Size:   604.07 MB

Deployment Sizes:
Flat disks:     16.00 GB
Sparse disks:   Unknown
Networks:
Name:        VM Network
Description: The VM Network network

Virtual Hardware:
Family:       vmx-04
Disk Types:   SCSI-lsilogic

Properties:
Key:         db_ip
Label:       IP address
Type:        ip:VM Network
Description: The IP address of the database server.

Key:         ws_ip
Label:       IP address
Type:        ip:VM Network
Description: The IP address of the Web server.

IP Allocation Policy:
Schemes:     ovfenv dhcp
Protocols:   IPv4
```

Using the VMware OVF Tool Machine Mode

6

This chapter includes the following topics:

About Machine Mode

You can use the --machineOutput option to run OVF Tool from another program or script. With the --machineOutput option, OVF Tool provides output in the following format:

```
STATUS-CODE details <blank line>.
```

OVF Tool inserts a blank line to signal the end of an operation. Each response line is prefixed with a plus (+) to avoid confusion with the terminating blank line. The last status it sends is always RESULT. OVF Tool sends all output, including errors and warnings, to standard output (stdout) so clients can listen on only one stream.

Table 6-1. Machine Mode Status

Status	Details	Description
PROBE	XML	Probe result with information about the source.
VALIDATEHOST	XML	Shows whether the VI target is compatible with the input arguments.
AUTHENTICATION	source/target/proxy server locator or fileName	Shows that authentication is required.
CERTIFICATE	Validate, Self-signed, or Failed validate	Signals that a certificate is present and shows the result of the validation.
MANIFEST	`Validate` or Failed validate	Signals that a manifest is present and shows the result of validation.
PROGRESS	Number (0-100)	Shows the percentage progress during upload and download.
TARGET_ID	Text, for example SugarCRM.ovf or vim.VirtualMachine:vm-415.	Shows the target ID after upload and download finishes.
RESULT	ERROR or SUCCESS	Always use as the last command sent.

Running Machine Output

You can run the --machineOutput option in different modes: Probe, Validate host, Import, and Export.

Running machineOutput in Probe Mode

When you run the machineOutput option in probe mode, OVF Tool reports the following status sequence:

1 AUTHENTICATION (zero or more)

2 PROBE (exactly one)

3 RESULT (exactly one)

To run the machineOutput option in probe mode, you run the following command.

```
ovftool.exe --machineOutput source_locator
```

For an example, see Output from Running machineOutput in Probe Mode.

Running machineOutput in Validate Host Mode

When you run machineOutput in validate host mode, OVF Tool reports the following status sequence:

1 AUTHENTICATION (zero or more)

2 VALIDATEHOST (exactly one)

3 PROGRESS (exactly one)

4 TARGET_ID (exactly one)

5 RESULT (exactly one)

To run the machineOutput option in validate host mode, you run the following command.

```
ovftool.exe --machineOutput --verifyOnly source_locator destination_locator
```

For an example, see Output from Running machineOutput in Validate Host Mode.

Running machineOutput in Import to vSphere Mode

When you run the machineOutput option in import mode, OVF Tool reports the following status sequence:

1 AUTHENTICATION (zero or more)

2 MANIFEST (zero or one)

3 CERTIFICATE (zero or one)

4 PROGRESS (one or more)

5 TARGET_ID (exactly one)

6 RESULT (exactly one)

To use machine mode to upload an OVF to vSphere, you run the following command.

```
ovftool.exe --machineOutput                                          \
            --acceptAllEulas                                         \
            --I:morefArgs                                            \
            --I:targetSessionTicket=<session ticket>                 \
            --net:<ovf netname>=vim.Network:<moref-id>               \
            --datastore=vim.Datastore:<moref-id>                     \
            --vmFolder=vim.Folder:<moref-id>                         \
            --deploymentOption=<value>                               \
            --diskMode=<value>                                       \
            --ipAllocationPolicy=<value>                             \
            --ipProtocol=<value>                                     \
            --name=<value> (optional)                                \
            --overwrite (optional)                                   \
            --powerOffTarget (optional)                              \
            --powerOn (optional)                                     \
            --prop:<key>=<value>                                     \
            <src URL or PATH>                                        \
             vi://<servername>?moref=vim.ResourcePool:<moref-id>
```

For an example, see Output from Running machineOutput in Import Mode.

Running the Machine Mode Export from vSphere Operation

When you run the machineOutput option in export mode, OVF Tool reports the following status sequence:

1 AUTHENTICATION (zero or more)

2 MANIFEST (zero or one)

3 CERTIFICATE (zero or one)

4 PROGRESS (one or more)

5 TARGET_ID (exactly one)

6 RESULT (exactly one)

To use machine mode to download an OVF from vSphere, you run the following command.

```
ovftool.exe --machineOutput                                          \
            --I:sourceSessionTicket=<session ticket>                 \
            -tt <OVA or OVF>                                         \
            -n=<name>                                                \
            --overwrite (optional)                                   \
            --powerOffSource (optional)                              \
            --chunkSize=<value> (optional)                           \
            --compress=<value> (optional)                            \
            vi://<servername>?moref=<type>:<moref-id>                \
            <directory>
```

The type value is either vim.VirtualMachine or vim.VirtualVApp.

When you specify --machineOutput, OVF Tool monitors STDIN, and cancels the operation if it reads the ABORT\n line in stdin.

For an example, see Output from Running machineOutput in Export Mode.

Example Output

You can run the OVF Tool machine mode `--machineOutput` option in probe mode, validate host mode, or import mode. In import and validate Host modes, `--machineOutput` is meant for third party program to use silently without any interactive prompt appearing in the standalone OVF tool command console. If your OVF file contains EULA information, you should add this option `--acceptAllEulas` so that the program doesn't pause to wait for acceptance of the license agreement. This section contains the following topics:

- Output from Running machineOutput in Probe Mode
- Output from Running machineOutput in Validate Host Mode
- Output from Running machineOutput in Import Mode
- Output from Running machineOutput in Export Mode

Output from Running machineOutput in Probe Mode

The following example shows the output of a --machineOutput PROBE operation on a file named LAMP.ovf.

Example: Output from Running machineOutput in Probe Mode

```
ovftool --machineOutput LAMP.ovf
PROBE
+ <probeResult>
+ <virtualApp>
+ true
+ </virtualApp>
+ <productInfo>
+ <name>
+ LAMP running PHP-Fusion
+ </name>
+ <productUrl>
+ http://example.com/ovf/1.0/LAMP/readme.txt
+ </productUrl>
+ <version>
+ 0.1
+ </version>
+ <fullVersion>
+
+ </fullVersion>
+ <vendor>
+ VMware
+ </vendor>
+ <vendorUrl>
```

```
+
+ </vendorUrl>
+ </productInfo>
+ <annotation>
+ This vApp offers the programming environment stack: Linux, Apache, MySQL and PHP programming
environment -- LAMP. More specifically the vApp contains a Database server running MySQL and Web
server VM running Apache2 and PHP.
+ </annotation>
+ <eulas>
+ <eula>
+
+    Eula for OVF
+
+ </eula>
+ </eulas>
+ <sizes>
+ <download>
+ 633412608
+ </download>
+ <flat>
+ 17179869184
+ </flat>
+ <sparse>
+ Unknown
+ </sparse>
+ </sizes>
+ <networks>
+ <network>
+ <name>
+ VM Network
+ </name>
+ <description>
+ The VM Network network
+ </description>
+ </network>
+ </networks>
+ <properties>
+ <property>
+ <classId>
+
+ </classId>
+ <key>
+ db_ip
+ </key>
+ <instanceId>
+
+ </instanceId>
+ <category>
+
+ </category>
+ <label>
+ IP address
+ </label>
+ <type>
+ ip:VM Network
```

```
+ </type>
+ <description>
+ The IP address of the database server.
+ </description>
+ <value>
+
+ </value>
+ </property>
+ <property>
+ <classId>
+
+ </classId>
+ <key>
+ ws_ip
+ </key>
+ <instanceId>
+
+ </instanceId>
+ <category>
+
+ </category>
+ <label>
+ IP address
+ </label>
+ <type>
+ ip:VM Network
+ </type>
+ <description>
+ The IP address of the Web server.
+ </description>
+ <value>
+
+ </value>
+ </property>
+ </properties>
+ <deploymentOptions>
+ </deploymentOptions>
+ <ipAllocationSchemes>
+ ovfenv,dhcp
+ </ipAllocationSchemes>
+ <ipProtocols>
+ IPv4
+ </ipProtocols>
+ </probeResult>

RESULT
+ SUCCESS
```

Output from Running machineOutput in Validate Host Mode

The following example shows the output of a --machineOutput VALIDATEHOST operation on file LAMP.ovf.

Example: Output from Running machineOutput in Validate Host Mode

`ovftool --machineOutput --acceptAllEulas --verifyOnly LAMP.ovf vi://myuser:mypassword@myvc.example.com/dc/host/myhost.example.com`

```
ovftool --machineOutput --acceptAllEulas --verifyOnly LAMP.ovf
vi://myuser:mypassword@myvc.example.com/dc/host/myhost.example.com
VALIDATEHOST
+ <supportedDiskProvisioning>
+ <type>
+ monolithicFlat
+ </type>
+ <type>
+ thin
+ </type>
+ <type>
+ thick
+ </type>
+ <type>
+ flat
+ </type>
+ <type>
+ eagerZeroedThick
+ </type>
+ </supportedDiskProvisioning>


PROGRESS
+ 0

TARGET_ID
+

RESULT
+ SUCCESS
```

Output from Running machineOutput in Import Mode

The following example shows the output of a --machineOutput import operation on a file named LAMP.ovf.

Example: Output from Running machineOutput in Import Mode

`ovftool --machineOutput --acceptAllEulas LAMP.ovf vi://myuser:mypassword@myvc.example.com/dc/host/myhost.example.com`

```
PROGRESS
+ 0
+ 1
+ 2
+ 3
....
```

```
+ 98
+ 99
+ 100

TARGET_ID
+ vim.VirtualApp:resgroup-v61

RESULT
+ SUCCESS
```

Output from Running machineOutput in Export Mode

The following example shows the output of a --machineOutput export operation on a file named LAMP.ovf.

Example: Output from Running machineOutput in Export Mode

```
ovftool -o --machineOutput --acceptAllEulas
vi://myuser:mypassword@myvc.example.com/dc/vm/LAMP /tmp/LAMP.ovf
```

```
PROGRESS
+ 0
+ 1
+ 2
+ 3
...
+ 98
+ 99
+ 100

TARGET_ID
+ /tmp/LAMP.ovf
RESULT
+ SUCCESS
```