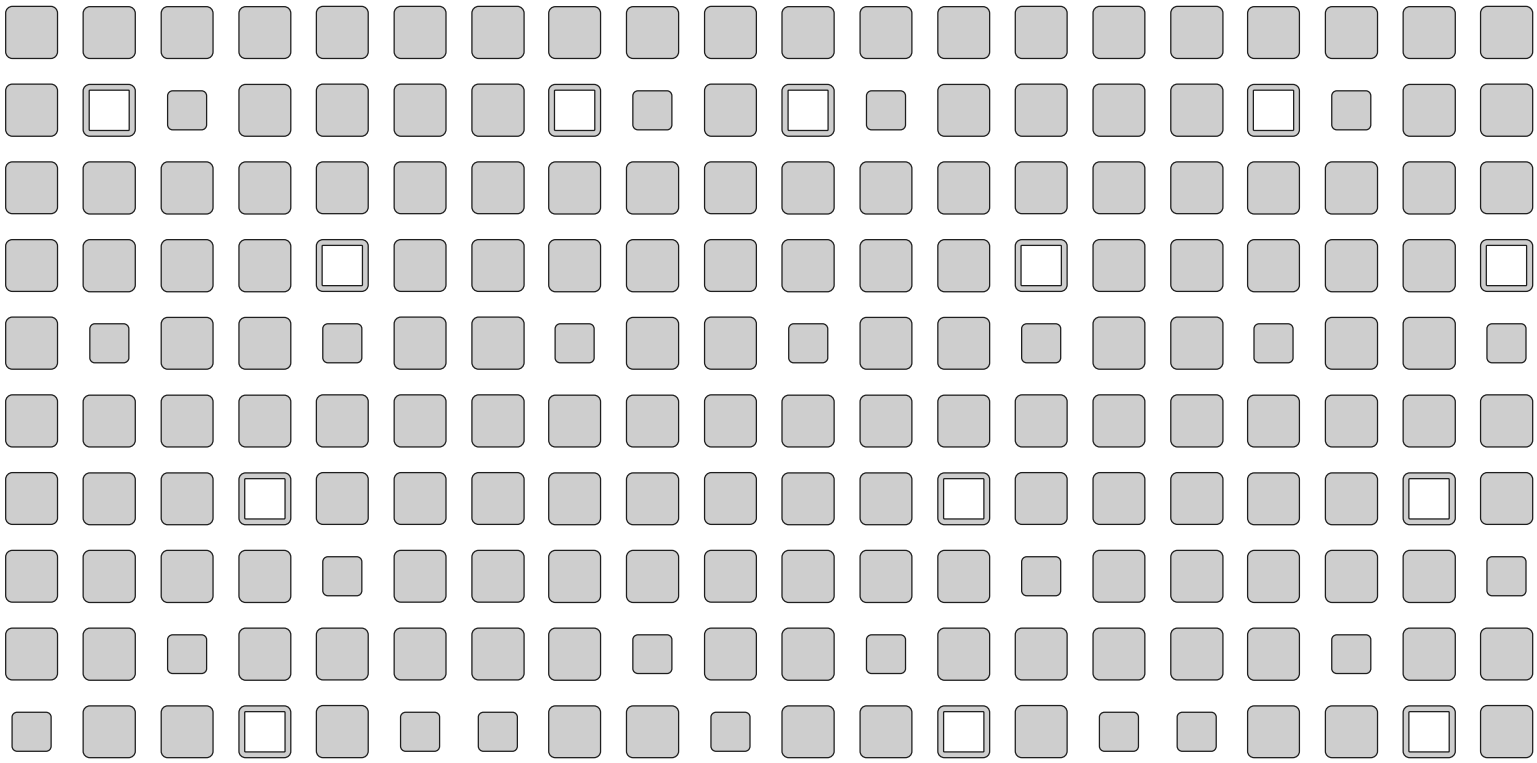
VERSION 1.4

Virtual Infrastructure SDK

Reference Guide



vmware®

WIZ, Inc. EXHIBIT - 1109
WIZ, Inc. v. Orca Security LTD.

VMware, Inc.

3145 Porter Drive
Palo Alto, CA 94304
www.vmware.com

Please note that you will always find the most up-to-date technical documentation on our Web site at http://www.vmware.com/support/.

The VMware Web site also provides the latest product updates.

Copyright © 1998-2005 VMware, Inc. All rights reserved. Protected by one or more of U.S. Patent Nos. 6,397,242, 6,496,847, 6,704,925, 6,711,672, 6,725,289, 6,735,601, 6,785,886, 6,789,156 and 6,795,966; patents pending. VMware is a registered trademark and the VMware "boxes" logo, GSX Server, ESX Server, Virtual SMP, VMotion and VMware ACE are trademarks of VMware, Inc. Microsoft, MS-DOS, Windows, Windows Server and Windows NT are registered trademarks of Microsoft Corporation. Linux is a registered trademark of Linus Torvalds. Novell and Netware are registered trademarks of Novell, Inc. All other marks and names mentioned herein may be trademarks of their respective companies. Revision 20051025 Version 1.4 Item: SDK-ENG-Q305-141

Table of Contents

Introducing VMware SDK 13
Overview of the Reference Guide 14
Using this Reference Guide 15
Intended Audience 15
Using Web Services 16
Overview of the VMware VirtualCenter Web Service 17
Element Management Operations 17
Virtual Computing Operations 18
Technical Support Resources 19
Standards and Reference Documentation 19

Data Model 21
Core Data Model 23
Host Machine Data Model 24
Virtual Machine Data Model 26
Performance Metric Data Model 29
Event Data Model 31
Task Data Model 32
Template Data Model 33
Guest Customization Data Model 34

Datatypes 37
AfterStartupTask 38
Autostart 38
Autostop 38
CPUControls (ESX Server Only) 39
CPUInfo 39
CPUPerf 40
CPUPerf2 (ESX Server Only) 40
Change 41
ChangeConflictFaultInfo 43
ChangeOp 43
ChangeReq 46
ChangeReqList 46
ConfigLimits 47
Container 47

ControllerConfigLimits 48
CustomPropertyList 48
DailyTask 49
DatastoreInfo 49
DatastoreInfoList 49
DiskAdapterType 50
DiskControllerType 50
DiskControls (ESX Server Only) 50
DiskDeviceAccessType 50
DiskDeviceInfo 51
DiskFileInfo 52
DiskKind 53
DiskPerf 53
DiskType 54
Event 54
EventArgInfo 55
EventCollection 55
EventCollector 55
EventDecl 55
EventDeclList 56
EventFilter 56
EventKind 57
Farm 57
FaultInfo 57
FaultKind 59
GenericDeviceInfo 60
GenericDevicePerf (ESX Server Only) 61
Guest Customization Datatypes 61
GuestInfo 68
GuestNetworkAdapter 69
GuestOSInfo 69
GuestOSMemoryInfo 70
HandleFaultInfo 71
HandleFaultKind 71
HandleFaultList 72
Hardware 72
Host 72
HostInfo 73

HostSpec 74
HostState 75
HourlyTask 75
Item 76
KeyedValue 76
Level 77
MemoryControls (ESX Server Only) 77
MemoryInfo 78
MemoryPerf 78
MemoryPerf2 (ESX Server Only) 78
MigrationInfo 79
MonthlyTask 80
NetworkControls (ESX Server Only) 81
NetworkInfo 81
NetworkPerf 81
OneTimeTask 82
PerfCollection 82
PerfCollector 82
PerfFilter 83
PerfSample 83
PerfSource 84
PerfSourceType 84
PerfStat 84
PerfStatType 85
PerfType 86
Permission 86
PermissionList 86
Property 86
RemovableDeviceInfo 87
ServiceStat 87
SystemInfo 88
Task 88
TaskRunState 89
TaskSchedule 90
TaskScheduleSpec 90
Template 91
TemplateSpec 91
Update 92

UpdateList 92
VHandleList 92
VMMemoryLimits 92
ViewContents 93
ViewInfo 93
VirtualCPUInfo 93
VirtualDisk 94
VirtualDiskDestination 94
VirtualDiskInfo 94
VirtualDiskMode 97
VirtualHardware 97
VirtualMachine 98
VirtualMachineCPUPerf (ESX Server Only) 98
VirtualMachineGroup 99
VirtualMachineInfo 99
VirtualMachineMemoryPerf 103
VirtualMachineMemoryPerf2 (ESX Server Only) 103
VirtualMachineSpec 104
VirtualMachineState 104
VirtualMachineTools 107
VirtualMachineToolsStatus 108
VirtualMemoryInfo 108
VirtualNetworkAdapter 108
VirtualNetworkAddressType 109
VirtualNetworkEmulation 110
VirtualNetworkInfo 110
VirtualNetworkMode 110
WeekDay 111
WeeklyTask 111
msgWaiting 112

Basic Operations 113
Login 115
Permissions 115
Input Message – LoginRequestMsg 115
Output Message – LoginResponseMsg 115
Fault 116

Logout 117
Permissions 117
Input Message – LogoutRequestMsg 117
Output Message – LogoutResponseMsg 117
Fault 117
ResolvePath 118
Permissions 118
Input Message — ResolvePathRequestMsg 118
Output Message — ResolvePathResponseMsg 118
Fault 118
GetContents 120
Permissions 120
Input Message – GetContentsRequestMsg 120
Output Message – GetContentsResponseMsg 120
Fault 122
GetInfo 123
Permissions 123
Input Message — GetInfoRequestMsg 123
Output Message — GetInfoResponseMsg 123
Fault 123

Element Management Operations 125
Performing Element Management Operations 127
Virtual Machine Namespace Management, Inventory and Discovery 127
Virtual Machine Power Operations 127
Virtual Machine Creation, Deletion and Configuration 127
Real-Time Performance Data Gathering 128
Event Notification 128
Virtual Machine State Transitions 129
Host State Transitions 130
StartVM 131
Permissions 131
Input Message — StartVMRequestMsg 131
Output Message — StartVMResponseMsg 131
Fault 132
AnswerVM 133
Permissions 133
Input Message — AnswerVMRequestMsg 133

Output Message — AnswerVMResponseMsg 133
Fault 134
StopVM 135
Permissions 135
Input Message — StopVMRequestMsg 135
Output Message — StopVMResponseMsg 135
Fault 136
ResetVM 137
Permissions 137
Input Message — ResetVMRequestMsg 137
Output Message — ResetVMResponseMsg 137
Fault 138
StopHost 139
Permissions 139
Input Message — StopHostRequestMsg 139
Output Message — StopHostResponseMsg 139
Fault 139
GetUpdates 141
Permissions 141
Input Message — GetUpdatesRequestMsg 141
Output Message — GetUpdatesResponseMsg 142
Fault 143
CancelGetUpdates 144
Permissions 144
Input Message — CancelGetUpdatesRequestMsg 144
Output Message — CancelGetUpdatesResponseMsg 144
Fault 144
PutUpdates 146
Permissions 146
Input Message — PutUpdatesRequestMsg 146
Output Message — PutUpdatesResponseMsg 147
Fault 147
Create 148
Permissions 148
Input Message — CreateRequestMsg 149
Output Message — CreateResponseMsg 153
Fault 153

CreateVirtualDisk 156
Permissions 156
Input Message — CreateVirtualDiskRequestMsg 156
Output Message — CreateVirtualDiskResponseMsg 157
Fault 157
Rename 158
Permissions 158
Input Message — RenameRequestMsg 158
Output Message — RenameResponseMsg 159
Fault 159
Delete 162
Permissions 162
Input Message — DeleteRequestMsg 162
Output Message — DeleteResponseMsg 162
Fault 163
QueryPerfData 164
Permissions 164
Input Message — QueryPerfDataRequestMsg 164
Output Message — QueryPerfDataResponseMsg 164
Fault 165
QueryPerfData2 166
Permissions 166
Input Message — QueryPerfData2RequestMsg 166
Output Message — QueryPerfData2ResponseMsg 166
Fault 167
ChangePermissions 168
Permissions 168
Input Message — ChangePermissionsRequestMsg 168
Output Message — ChangePermissionsResponseMsg 168
Fault 168
SnapshotVM 170
Permissions 170
Input Message — SnapshotVMRequestMsg 170
Output Message — SnapshotVMResponseMsg 171
Fault 171
ConsolidateVM 172
Permissions 172
Input Message — ConsolidateVMRequestMsg 172

Output Message — ConsolidateVMResponseMsg 172
Fault 173
RevertVM 174
Permissions 174
Input Message — RevertVMRequestMsg 174
Output Message — RevertVMResponseMsg 174
Fault 175

Virtual Computing Operations 177
Performing Virtual Computing Operations 178
Virtual Machine Migration, including VMotion 178
Virtual Machines Are Managed Independently of Hosts 178
Virtual Machine Provisioning and Deployment 178
Task Scheduling 178
EnableHost 179
Permissions 179
Input Message — EnableHostRequestMsg 179
Output Message — EnableHostResponseMsg 179
Fault 180
DisableHost 181
Permissions 181
Input Message — DisableHostRequestMsg 181
Output Message — DisableHostResponseMsg 181
Fault 181
MoveVM 182
Permissions 182
Input Message — MoveVMRequestMsg 183
Output Message — MoveVMResponseMsg 183
Fault 184
MigrateVM 185
Permissions 185
Input Message — MigrateVMRequestMsg 185
Output Message — MigrateVMResponseMsg 185
Fault 186
CloneVM 187
Permissions 187
Input Message — CloneVMRequestMsg 187
Output Message — CloneVMResponseMsg 188

Fault 189
CreateTemplate 190
Permissions 190
Input Message — CreateTemplateRequestMsg 190
Output Message — CreateTemplateResponseMsg 190
Fault 191
RunTask 192
Permissions 192
Input Message — RunTaskRequestMsg 192
Output Message — RunTaskResponseMsg 192
Fault 192
EndTask 193
Permissions 193
Input Message — EndTaskRequestMsg 193
Output Message — EndTaskResponseMsg 193
Fault 193

Sample Code 195
Copyright Information 196

Glossary 197

Virtual Machine Configuration Variables 201

VMware VirtualCenter Events 203
VirtualCenter Event Messages 204
ESX Server Messages About the Host or Virtual Machine 204
Messages About the User 204
Host Connection Messages 204
Virtual Machine Messages 205
Cloning and Migration Messages 206
Task Messages 207
Alarm Messages 207
Storage Message 208

Performance Statistics 209
Calculating Performance Statistics 210
CPU Statistics on the Host Summary Page 210
Memory Performance Statistics 211
Disk, Generic Device, and Network Performance Statistics 211

Valid TimeZone Indexes 213

Revision History 217

Index 219

CHAPTER 1

Introducing VMware SDK

The goal of the VMware® Virtual Infrastructure SDK (VMware SDK) is to provide Independent Software Vendors (ISVs) a seamless way to integrate management of VMware products into their data center management solutions. ISVs include systems management vendors, enterprise management framework vendors, imaging and provisioning vendors, storage management vendors, and so on.

With the VMware SDK, Independent Software Vendors can:

- Automate provisioning of new virtual machines.
- Support discovery and inventory functions in virtual environments.
- Access real-time performance data.
- Control all virtual machine-related operations from within the systems management framework, including configuration changes and migrations.
- Trigger virtual machine operations, specifically migrations and failover operations, based on application level monitoring.

Overview of the Reference Guide

The purpose of this reference guide is to help you familiarize yourself with the logical structure of the VMware VirtualCenter Web Service, the datatypes present in `vma.wsdl`, and the Web service operations. Developers who use this manual should be familiar with the operation and management of VMware® VirtualCenter, VMware® ESX Server™, VMware® GSX Server™ and other VMware products.

We discuss the following topics in this reference guide:

- Data Model on page 21

  This chapter describes the different data models that comprise the VMware VirtualCenter Web Service.

- Datatypes on page 37

  This chapter provides a complete list of the datatypes in the VMware SDK product, including a brief description of each field.

- Basic Operations on page 113

  This chapter describes the basic API operations in the VMware SDK product: Login, Logout, ResolvePath, and GetContents.

- Element Management Operations on page 125

  This chapter describes API operations on individual virtual machines and hosts.

- Virtual Computing Operations on page 177

  This chapter describes API operations that enable a higher-level functionality for managing virtual machines across a farm of physical resources.

- Sample Code on page 195

  This chapter presents the copyright for the sample code included with the VMware SDK product.

- Virtual Machine Configuration Variables on page 201

  This appendix lists the configuration variables that describe the virtual machine and its guest operating system.

Using this Reference Guide

This *Virtual Infrastructure SDK Reference Guide* is a companion book to the *Virtual Infrastructure SDK Programming Guide*.

This *Virtual Infrastructure SDK Reference Guide* contains a description of the data models, or logical structure of the VMware VirtualCenter Web Service. This is followed by a list of the datatypes present in `vma.wsdl`. Finally, this guide contains a comprehensive list of all the Web service operations, including its input message, its output message, and any fault messages.

The *Virtual Infrastructure SDK Programming Guide* includes a detailed description of VMware SDK concepts and how clients interact with the Web service. We discuss how to create a simple client application, then discuss basic and advanced programming concepts to help you build your client application. Finally, we discuss the different developer environments you may use, followed by a description of the sample and reference applications supplied in the VMware SDK package.

Note: Be sure to read the first two chapters in the programming guide before using the VMware SDK product. These two chapters contain information that helps you understand the SDK architecture and terminology and will enhance your use of this reference guide.

Intended Audience

This reference guide is written for programmers that are familiar with Web services concepts and principles. Readers of this manual should be comfortable with developing system administration and system monitoring programs and be familiar with general debugging techniques. In addition, developers who use this manual should be familiar with the operation and management of VMware VirtualCenter and ESX Server.

Note: In this release, the VMware SDK supports VMware VirtualCenter 1.2, and ESX Server 2.0.1, 2.1.x, and 2.5, and GSX Server 3.1.

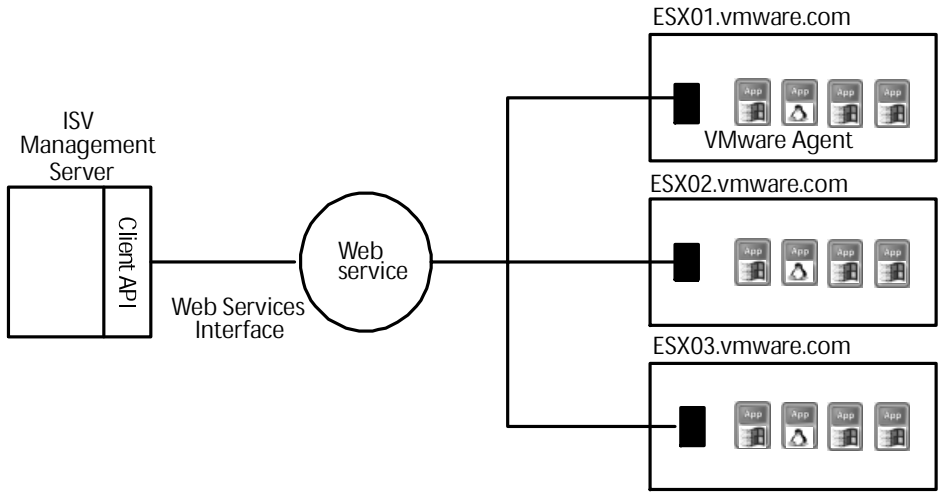
Using Web Services

Web services are applications whose logic and functions are accessible using standard Internet protocols and data formats such as Extensible Markup Language (XML) over (Secure) Hypertext Transfer Protocol (HTTPS), and SOAP (Simple Object Access Protocol). A Web service interface is defined strictly in terms of the messages that the service accepts and generates. VMware provides a Web service interface to ISVs for the following reasons:

- Industry Support — ISVs and vendors are supporting the Web services movement. The SOAP framework is designed to be independent of any particular programming model and other implementation-specific semantics. Clients or applications using a Web service can be implemented on any platform, in any programming language, as long as they can create and consume messages defined for the service interface.
- Security — All client access to the Web service uses HTTPS, ensuring that management traffic is secure and encrypted.
- Remote Operations — With appropriate firewall rules and port forwarding, it is possible to manage VMware products remotely.
- Interoperability — VMware uses the SOAP and WSDL (Web Services Description Language) standards, so developers can program in their favorite environment, such as Java or Microsoft® Visual Studio® .NET.
- Ubiquity — Web services communicate using HTTP and XML. Any connected device that supports these technologies can both host and access Web services.
- Low Barrier to Entry — The concepts behind Web services are easy to understand, and developers can quickly create and deploy them using many tool kits available on the Web.

Overview of the VMware VirtualCenter Web Service

The following diagram provides an overview of how you can use the VMware SDK to manage your virtual machines:



The VMware SDK comprises the following:

- Web services API to clients for managing virtual machines running on one or more hosts.
- Documentation, including this Reference Guide and the *Virtual Infrastructure SDK Programming Guide*.
- Sample code that showcases the powerful functionality that is available to the clients.

The Web services API provides access to the following functionality:

- Element Management — Basic virtual machine management functionality within the realm of a physical host machine.
- Virtual Computing — virtual machine management functionality across host machines.

Element Management Operations

The VMware SDK provides the following element management operations:

- Virtual machine configuration (attaching to physical resources, modifying resource shares, and so on)
- Virtual machine power operations and state management (power on/off, suspend, reset, and so on)
- Virtual machine creation and deletion

- Virtual machine inventory
- Virtual disk management
- Collection of host and virtual machine performance data
- Discovery and event notification

Virtual Computing Operations

The VMware SDK provides the following virtual computing operations:

- Virtual machine provisioning and deployment
- Virtual machine migration, including VMotion™
- Virtual machines managed independently of underlying physical hosts

Technical Support Resources

Standards and Reference Documentation

Refer to the following for additional information.

- VMware SDK — *www.vmware.com/support/developer/vc-sdk*
- VMware ESX Server — *www.vmware.com/products/server/esx_features.html*
- VMware VirtualCenter — *www.vmware.com/products/vmanage/vc_features.html*
- W3C SOAP 1.1 Specifications — *www.w3.org/TR/SOAP*
- XML Schema — *www.w3.org/2001/XMLSchema*
- HTTPS (SSL v3) — *wp.netscape.com/eng/ssl3/ssl-toc.html*
- WSDL 1.1 — *www.w3.org/TR/wsdl*
- HTTP 1.1 — *www.ietf.org/rfc/rfc2616.txt*
- XML 1.0 — *www.w3.org/TR/REC-xml*

CHAPTER 2

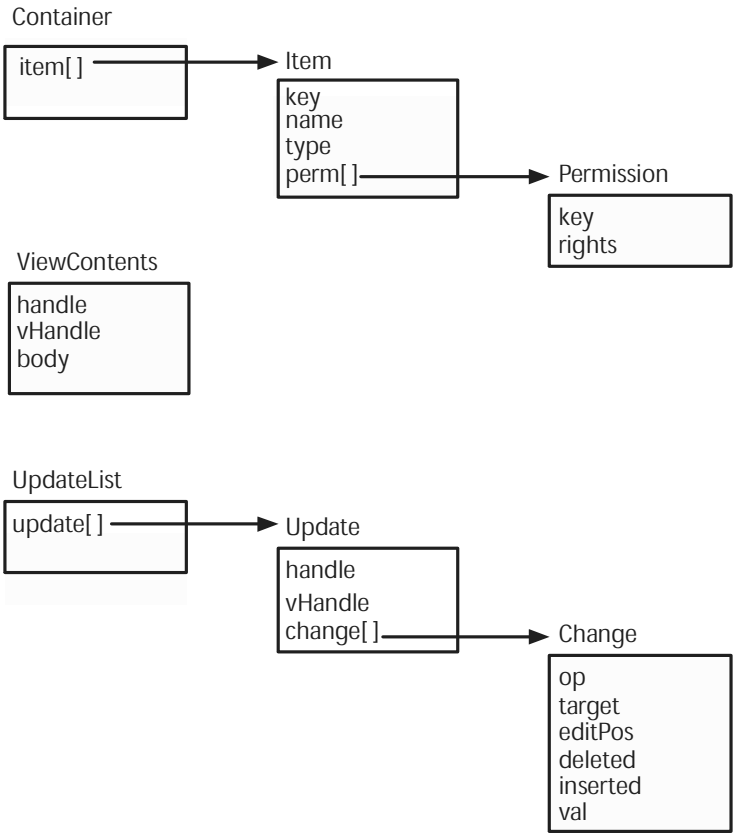
Data Model

The following sections discuss the VMware VirtualCenter Web Service data models; that is, the logical structure of this Web service. The details of each of the datatypes, along with the description of each of the fields, are given in the following sections. The data models are organized in the following major groups:

- Core Data Model — Traverses the hierarchy exported by the Web service.
- Host Machine Data Model — Describes the host machine configuration, including whether or not the machine is a non-uniform memory architecture (NUMA) machine.
- Virtual Machine Data Model — Describes the virtual machine configuration and sharing of the host machine.
- Performance Metric Data Model — Describes the performance metric of interest to the client, that is received from the Web service.
- Event Data Model — Describes the events that are generated on the host and virtual machines.

- Task Data Model — Describes the various tasks that may be automatically created or started by the client.
- Template Data Model — Describes the templates that may be used to create new virtual machines.
- Guest Customization Data Model — Describes the guest customization specification that may be used to customize newly created virtual machines.

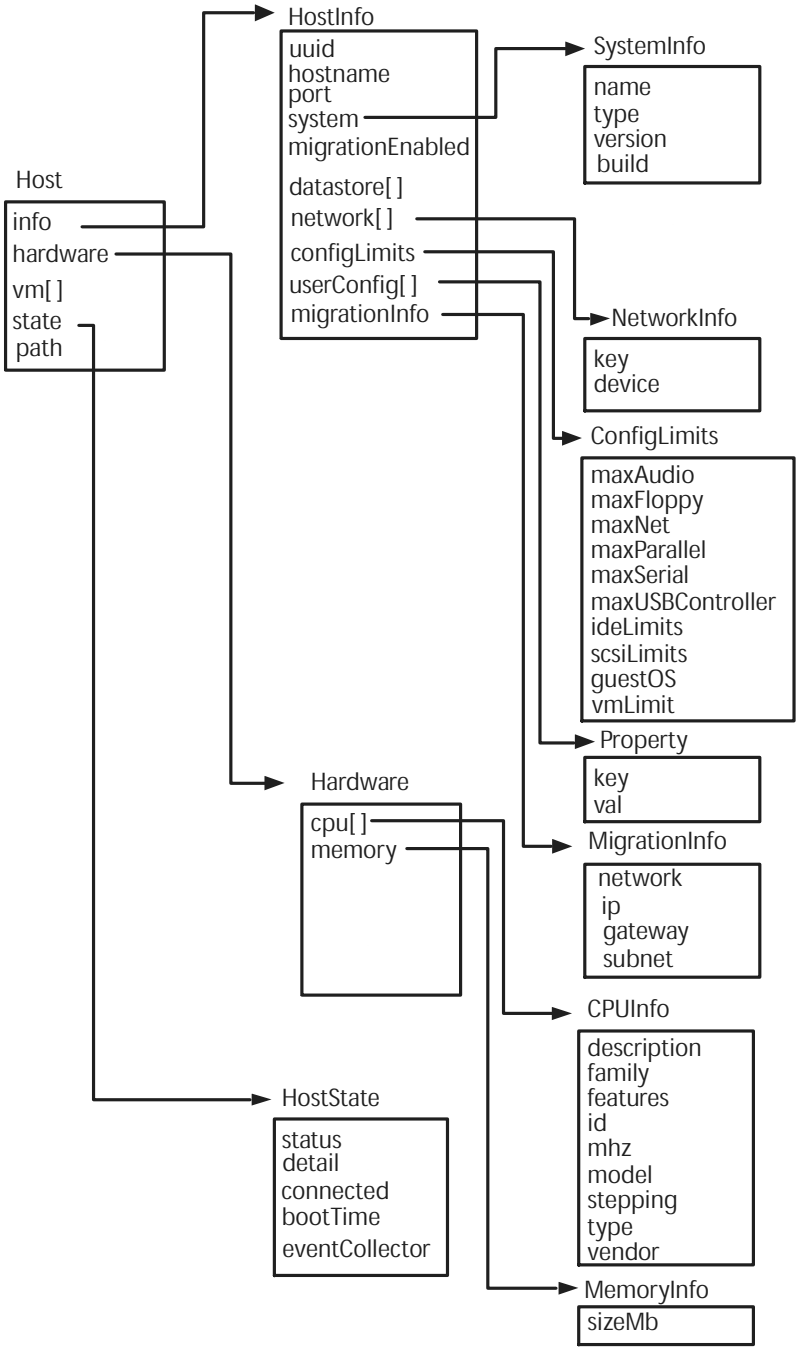
Core Data Model



The preceding figure shows the basic datatypes exported by the Web service:

- ViewContents — This datatype comprises the return value of a GetContents operation on a client specified handle. It consists of a body that can be of type `xsd:anyType`.
- Update — This datatype comprises the list of changes for the vHandle that is of interest to the client. The details of Change indicate the actual changes that have taken place.
- Container — This datatype comprises the Container datatype. It consists of a list of Items, each of which has a key, name, type, and access control list associated with it. The key contains the handle for the object that is identified by the name field.

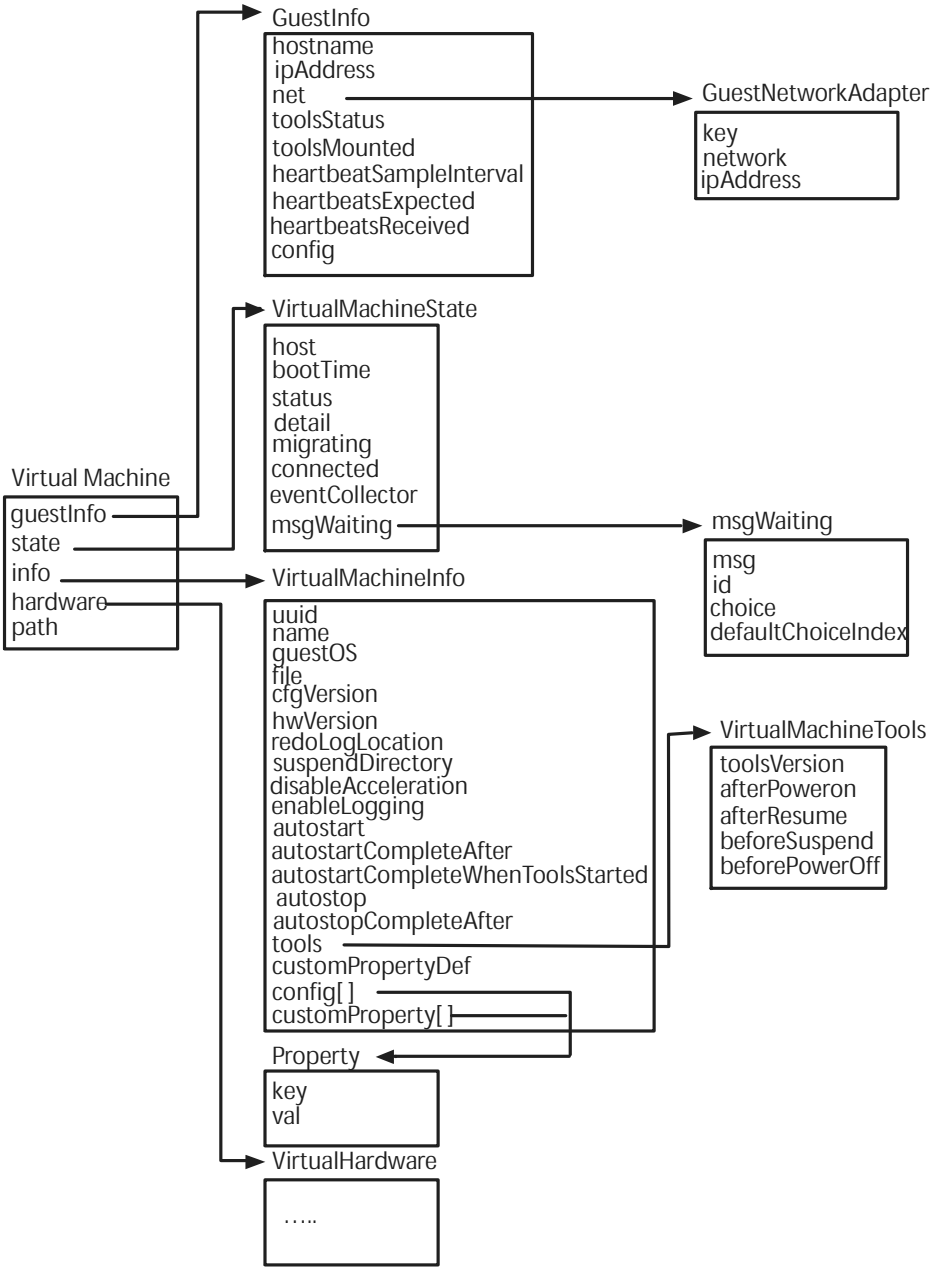
Host Machine Data Model



The preceding figure shows the host datatype exported by the Web service.

- Host — Describes the host machine, including references to the virtual machines on the host.
- HostInfo — Describes the basic host information.
- HostState — Describes the state of the host. Clients may compute the uptime using the `bootTime` value in the HostState structure.
- Hardware — Lists all the CPU and memory attached to the host.
- SystemInfo — Detailed information on the host operating system.
- VolumeInfo — Provides information about raw devices, usually (disk) volumes.
- NetworkInfo — Provides information about the host network interface.
- ConfigLimits — Provides the minimum and maximum limits for devices and supported guest operating systems.
- MigrationInfo — Provides information about a virtual machine migration.
- CPUInfo — Detailed information on the processors.
- MemoryInfo — Provides information about the memory in the host.

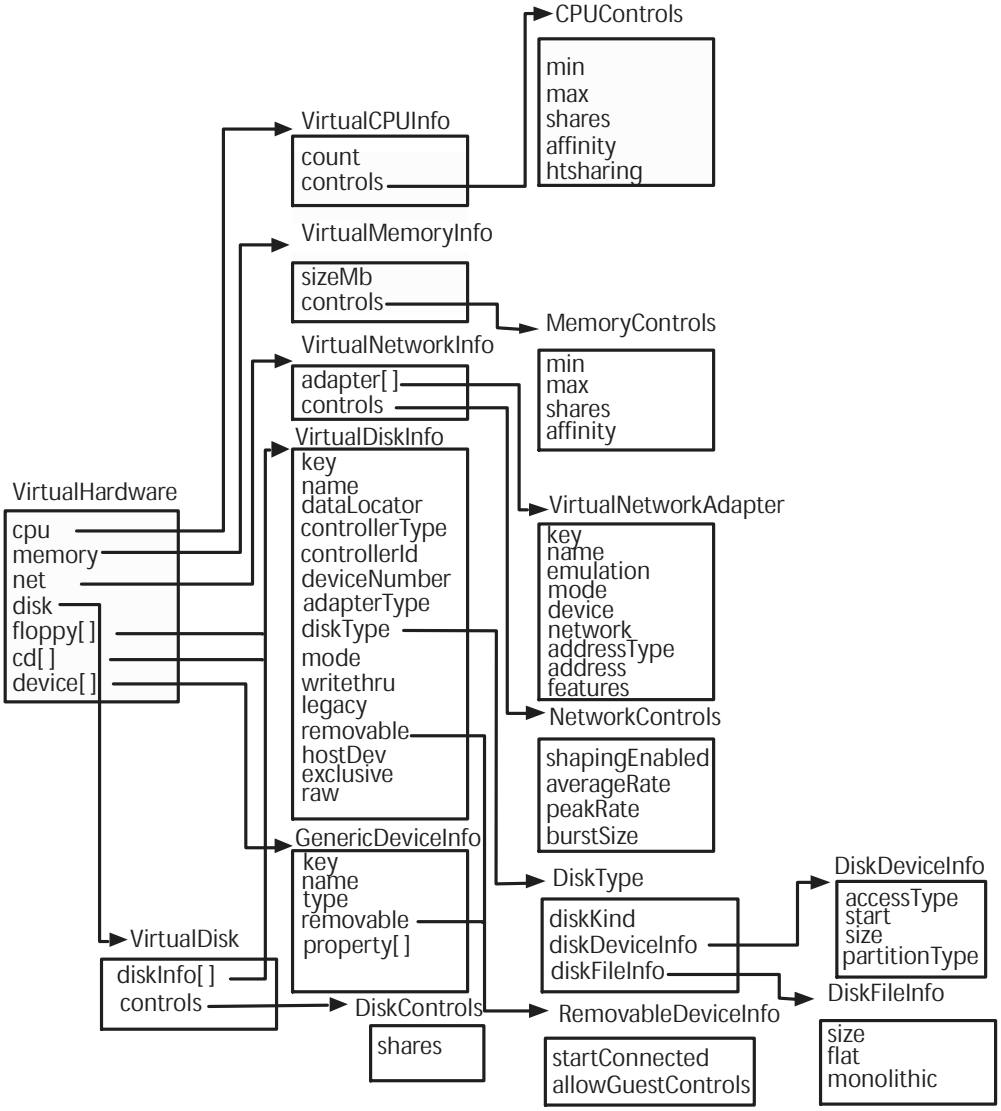
Virtual Machine Data Model



The preceding figure shows the Virtual Machine data model exported by the Web service.

- VirtualMachine — Each virtual machine has a state and a description of the virtual hardware.
- GuestInfo

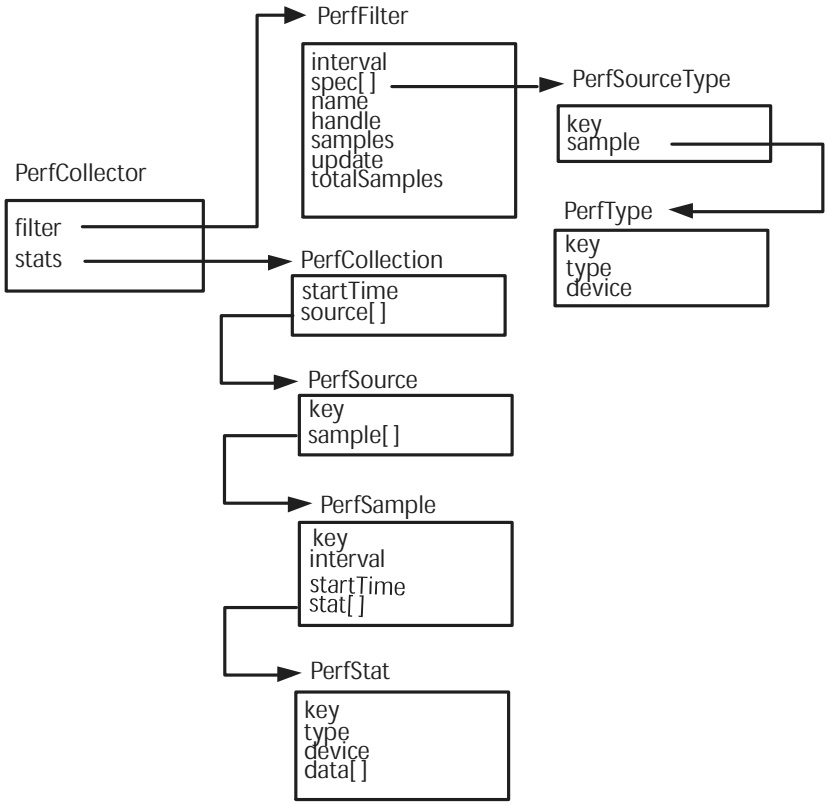
- VirtualMachineState — Provides detailed information on the current state of the virtual machine.
- VirtualMachineInfo — The `config` and `guestInfo` parameters have all of the configuration parameters for the virtual machine.
- VirtualMachineTools — Contains information about VMware Tools. This includes the scripts that run at different events (state changes) in the virtual machine life cycle.



The preceding diagram illustrates the VirtualHardware datatype for the Virtual Machine data model.

- VirtualHardware — Equivalent of the Hardware entry for the host machine. It describes all of the virtual hardware attached to a virtual machine.
- VirtualDisk — Provides resource management and configuration information for the disks associated with the virtual machine.
- VirtualCPUInfo — Provides resource management configuration information for the CPU associated with the virtual machine.
- VirtualMemoryInfo — Provides resource management configuration information for the memory associated with the virtual machine.
- VirtualNetworkInfo — Provides resource management and other configuration information for the network cards associated with the virtual machine.
- VirtualDiskInfo — Provides configuration information for a single virtual disk.
- CPUControls — Provides resource management configuration information for the CPUs in a virtual machine.
- MemoryControls — Provides resource management configuration information for the memory associated with the virtual machine.
- VirtualNetworkAdapter — Provides configuration information for the NICs associated with the virtual machine.
- NetworkControls — Provides information about network traffic shaping.
- DiskType — Provides information about the structure of a virtual disk that you are adding to a virtual machine.
- GenericDeviceInfo – Provides information about generic (non-disk and non-network) devices.
- RemovableDeviceInfo – Describes the parameters for devices that may be disconnected from the virtual machine.

Performance Metric Data Model

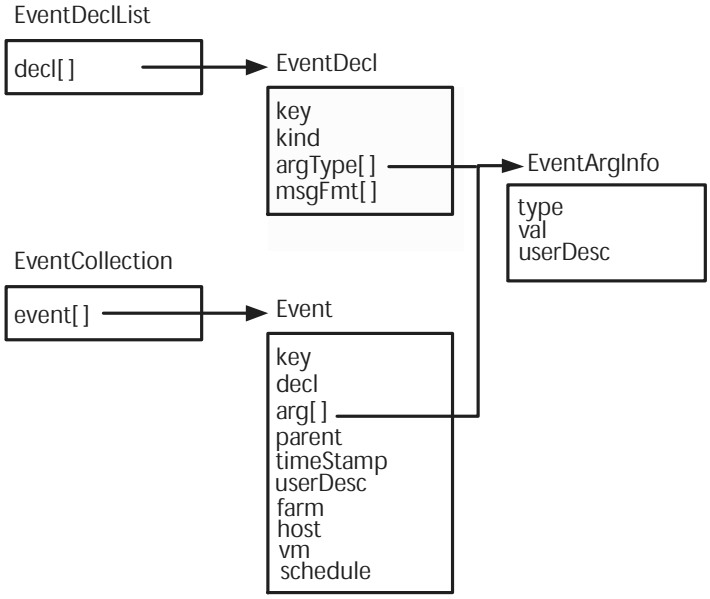


The preceding figure shows the performance metrics and resource utilization information data model exported by the Web service. This applies to virtual machines and to the host machine.

- PerfCollector — Specifies a collector of performance statistics. The filter specifies the sample interval and the performance statistics that are collected. Stats refers to the actual performance statistics that are collected.
- PerfFilter — Specifies the time interval between samples (interval), the type of performance statistics that are collected (spec), the name of the PerfCollector (name), the handle of the parent collector (handle) if there is one, the number of samples provided by the PerfCollector (samples), the number of new samples before beginning an update (update), and the available historical samples (totalSamples).
- PerfCollection — Specifies the data that is reported by a performance collector, including the start time for sample collection, and the list of performance statistics from various sources.
- PerfSource — Specifies performance statistics from a single source.
- PerfSample — Specifies performance statistics from a single source for a single sample.

- PerfStat — Specifies a performance statistic for a specified element (for example, a CPU) from a single source (a host or a virtual machine).
- PerfSourceType — Describes the specification of performance statistics from various sources.
- PerfType — Describes the specification of a particular statistic that is gathered from a particular (single) source.

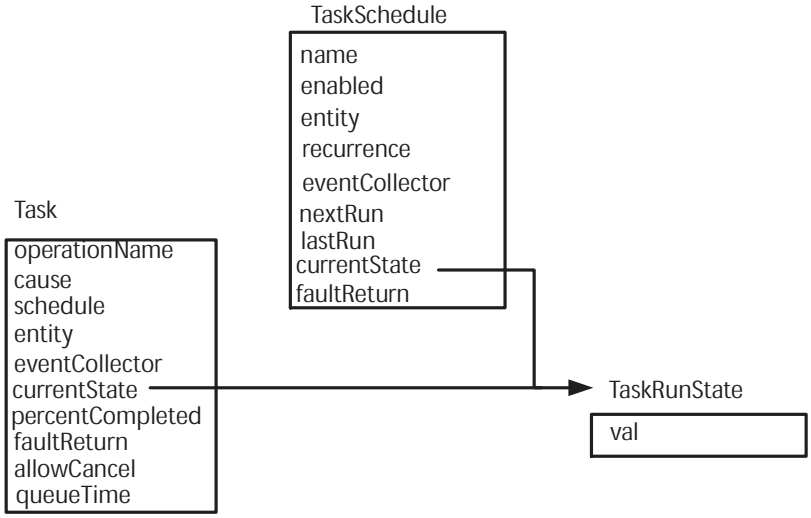
Event Data Model



The preceding figure shows the event logs abstraction exported by the VMware VirtualCenter Web Service.

- EventDeclList — Describes all event declarations.
- EventDecl — Describes an individual event declaration. An event declaration is the type of event (alert, error, warning, info, or user) along with its name, and its message format.
- EventArgInfo — Describes the arguments for an event.
- EventCollection — Specifies all the available events.
- Event — Describes an individual event.

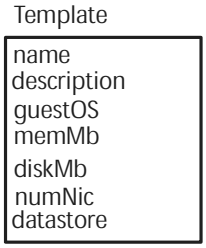
Task Data Model



The preceding figure shows the tasks abstraction exported by the VMware VirtualCenter Web Service. Clients may start a task and check the status. Tasks may be automatically started based on a client operation request if the task is expected to take a long time. Alternatively, clients may also schedule tasks.

- Task — Describes a running or completed task, including its name, any task parameters, the current task status, how close the task is to completion, and its return value, if any.
- TaskSchedule — Describes the schedule for a task, including the schedule name, the task, the recurrence of the task, the last time the task was run, and the next time the task will run.

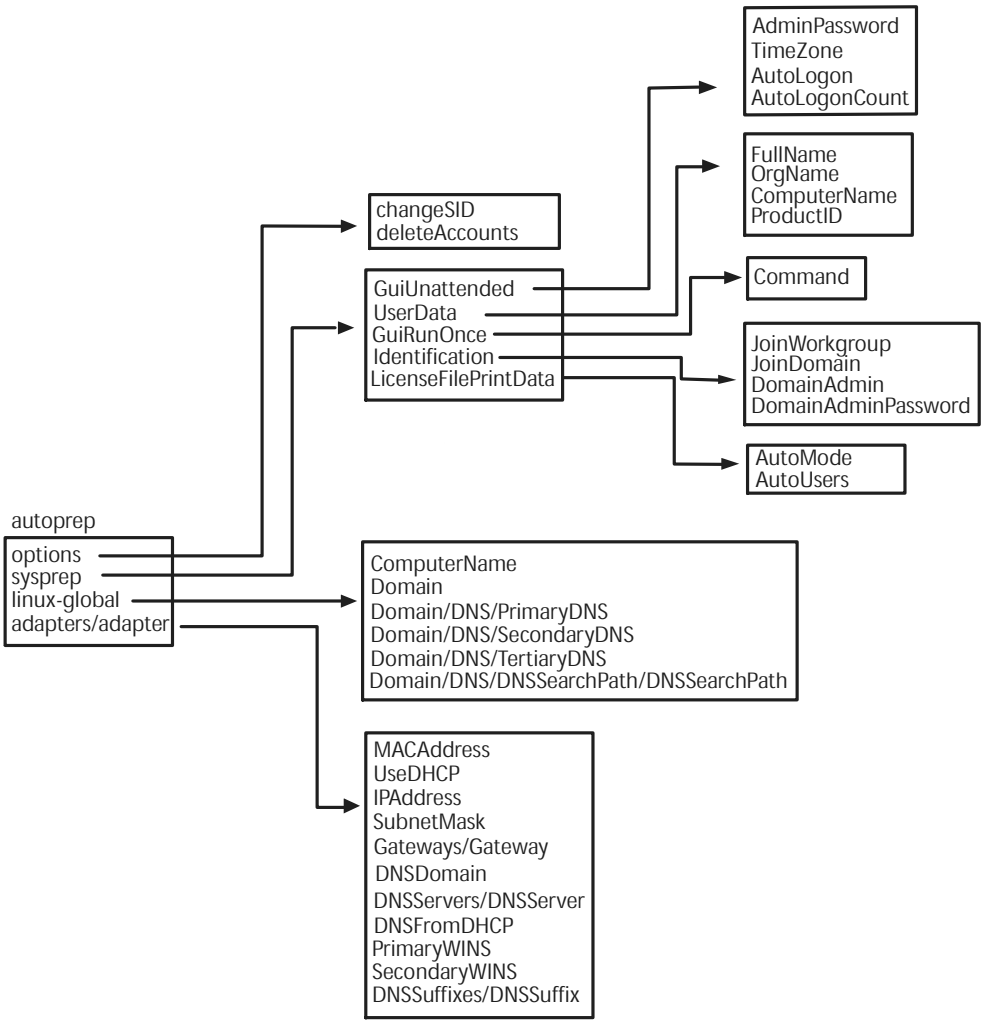
Template Data Model



The preceding figure shows the template data model exported by the VMware VirtualCenter Web Service. This describes a template for a virtual machine, that may be customized to create new virtual machines.

- Template — Describes a template used to create a virtual machine, including some system information and sizes.

Guest Customization Data Model



The preceding figure shows the guest customization data model exported by the Web service.

- options — Options that describe the entire customization process.
- sysprep — Customization operations that are implemented by Microsoft's sysprep automated deployment tool including the virtual machine's identity, regional settings, and the serial number.
- linux-global — Parameters that are relevant to a customization of a Linux guest operating system including basic network parameters, the host name for the virtual machine, and its domain name.

- adapters/adapter — Fields that customize network parameters to a specific virtual network adapter and are used both by Windows and Linux guest operating systems.

CHAPTER 3

Datatypes

This chapter provides a detailed description of each datatype in the Web Services Description Language (WSDL), listed in alphabetical order. To gain an understanding of the datatypes in a hierarchical order, first look at the descriptions of the types Container, Farm, and VirtualMachineGroup, and then follow references to other types from these types.

These datatypes reflect information that is sent from the client to the Web service as well as vice versa, from the Web service to the client (typically responses to GetContents, GetUpdates, QueryPerfData, and so on).

Each datatype is presented as a table. Each row of the table describes a field of the datatype. The columns in the table specify the name of the field, its type, and a description. Where applicable, some datatype tables have a column that indicates which fields may changed through a call to PutUpdates (Modifiable by PutUpdates).

Some of the parameters in the datatype tables are marked as "Optional". These parameters are not required and may or may not be present in a call to the Web service (when the datatype is modifiable by PutUpdates) or in a response from the Web service to the client.

Note: Datatype fields cannot be modified when a virtual machine is in the suspended state. Other fields can only be modified when the virtual machine is in the powered-off state (and not when the virtual machine is running).

We note these exceptions in the following tables, where necessary. All other datatype fields that are marked as modifiable, can only be changed when a virtual machine is either in the powered-on or powered-off states.

Note: Some of these datatypes are only supported on ESX Server, and are noted where appropriate. If there is no specific notation, then the datatype is supported on both ESX Server and GSX Server.

Note: Some of these tables are quite large and extend over several pages. When viewing a specific datatype, be sure to check whether or not the table extends onto the next page.

AfterStartupTask

AfterStartupTask describes a task scheduled to run after VirtualCenter starts up.

Name	Type	Description
delay	xsd:int	The task is scheduled to start this number of minutes after VirtualCenter starts up.

Autostart

Autostart allows a virtual machine to be automatically started when the host, on which the virtual machine resides, is powered on.

Name	Type	Description
val	xsd:string	Takes one of the following values: • none • poweron • softpoweron

Autostop

Autostop allows a virtual machine to be automatically stopped or suspended when the host, on which the virtual machine resides, is powered off.

Name	Type	Description
val	xsd:string	Takes one of the following values: • poweroff • softpoweroff • suspend • softsuspend

CPUControls (ESX Server Only)

CPUControls describes resource management for the virtual CPU.

Name	Type	Description	Modifiable by PutUpdates
min	xsd:int	(Optional) Minimum processor percentage for a virtual machine in [0, number of virtual CPUs * 100].	Yes
max	xsd:int	(Optional) Maximum processor percentage for a virtual machine in [0, number of virtual CPUs * 100].	Yes
shares	xsd:string	CPU shares allocation, number in [1, 100000], or special named value such as "low", "normal", or "high".	Yes
affinity	xsd:string	(Optional) Processor affinity set, specified as a comma-separated list of physical CPU numbers like "0,2,3" or special name values such as "default" or "all". This affinity set applies to all the virtual CPUs in a virtual machine.	Yes
htsharing	xsd:string	(Optional) Allows use of hyper-threading (HT), when available. Values are one of the following: "any" (allow any HT sharing), "none" (disallow HT sharing), or "internal" (allow HT sharing only with sibling VCPUs).	Yes

CPUInfo

CPUInfo provides detailed information about a physical processor. There is one CPUInfo structure for each processor on a host.

Name	Type	Description
key	xsd:string	Unique identifier for each CPUInfo structure in the array.
description	xsd:string	Description for display purposes.

Name	Type	Description
family	xsd:int	Family number of processor.
features	xsd:int	Bit mask containing CPU features. To determine if the CPU is hyperthreading-capable, check whether or not bit 28 of the 32 bit features flag is set. The bit mask to check bit 28 is `0x10000000`.
id	xsd:int	Unique identifier.
mhz	xsd:int	Frequency.
model	xsd:int	Model number of processor.
stepping	xsd:int	Stepping number of processor.
type	xsd:int	Processor type.
vendor	xsd:string	Vendor for the processor.

CPUPerf

CPUPerf describes the utilization of a virtual CPU or a physical CPU.

Name	Type	Description	Applies to
pcpu	xsd:int	(Optional) Physical CPU where this virtual CPU last executed.	ESX Server only
used	xsd:long	(Optional) Processor time consumed by each CPU, in milliseconds	ESX Server GSX Server
system	xsd:long	(Optional) System time consumed by each CPU, in milliseconds. Note that "system" is included in "used".	ESX Server only
wait	xsd:long	(Optional) Wait time for each CPU, in milliseconds.	ESX Server only

CPUPerf2 (ESX Server Only)

CPUPerf2 augments the CPUPerf datatype. It includes an additional optional field, ready.

Note: This new performance statistic field will only be available to the client by using the new QueryPerfData2 operation.

Name	Type	Description
pcpu	xsd:int	(Optional) Physical CPU where this virtual CPU last executed.
used	xsd:long	(Optional) Processor time consumed by each CPU, in milliseconds

Name	Type	Description
system	xsd:long	(Optional) System time consumed by each CPU, in milliseconds. Note that "system" is included in "used".
wait	xsd:long	(Optional) Wait time for each CPU, in milliseconds.
ready	xsd:long	(Optional) Ready time for each CPU, in milliseconds. A virtual CPU is "ready" when it is runnable but not currently scheduled.

Change

The Change datatype is the central mechanism by which a client and the Web service communicate about changes to objects and changes to interior nodes of objects. The client learns about such changes from the Web service by using the GetUpdates operation and sends any changes it wishes to make to the Web service by using the PutUpdates operation.

GetUpdates returns an UpdateList, which is an array of the Update datatype. Each Update contains the handle and new vHandle of an object that has changed, along with a set of changes to the object. Each change in the set is of the Change datatype. This mechanism allows GetUpdates to report multiple changes on more than one object with a single call.

PutUpdates is passed a ChangeReqList as its input parameter. This type is an array of the ChangeReq datatype where the ChangeReq contains a handle (unversioned PutUpdates call) or vHandle (versioned PutUpdates call) and a set of changes for the object. Each change in the set is of the Change datatype. This interface allows a single PutUpdates call to perform multiple changes on multiple objects. The current implementation of the Web service, however, allows only one object to be updated at a time although multiple changes can be made to that object by using one PutUpdates call.

Change Targets

There are two broad categories of targets that clients can use to communicate changes by using the Change datatype: leaf values and composites.

- Leaf value — An example of a leaf value is the value "My Great Virtual Machine" for the "name" field of the VirtualMachineInfo datatype. Leaf values correspond to text nodes within an XML document. This includes xsd:string, xsd:int, xsd:long and other similar primitive datatypes.
- Composites — A composite or aggregate is an entire object such as an object of type Item, Host, or VirtualMachine, or an interior node of such an object such as the VirtualNetworkAdapter inside the VirtualMachine datatype.

Arrays — Arrays are sets of composites that require an index or key value to select a specific composite from the set. The "adapter" node of a VirtualNetworkInfo object is an example of an array, where each element of the array is of type VirtualNetworkAdapter.

The VMware VirtualCenter Web Service API supports two kinds of arrays: indexed arrays and key-based arrays.

- Indexed Arrays — Clients access indexed arrays by an index integer. An example of an index-based array is the virtual machine handle array in a Host. For example, `vm[0]` in this array is a virtual machine handle in a Host's list of virtual machines.

  Note: The Web service does not have any indexed arrays that can be updated by a Web service client. All arrays that may be updated from the client are key-based arrays.

- Key-based Arrays — Key-based arrays always have a string field named "key" in each array element. The value of the key field is generated by the Web service and is unique for each array element. Key-based arrays are similar to associative arrays in languages such as Perl, that support arrays indexed by string values instead of by integers.

  Key-based arrays are very similar to indexed arrays except that the key is a string. The syntax for specifying an element using a key is `array["key"]` instead of `array[0]`. An example of a key-based array target is `hardware/net/adapter["#_nic0"]`, where adapter is a key-based array of type VirtualNetworkAdapter.

Note: Arrays can be the final target of a change or they can occur within the target expression. For example, both `hardware/net/adapter["#_nic0"]` and `hardware/net/adapter["#_nic0"]/name` are valid targets.

Target expressions such as `hardware/net/adapter["#_nic0"]` must be parsed by the client (and Web service) to extract the array key or index. In this case, the client has to parse this expression in order to extract the key value `#_nic0` for this element of the adapter array.

Fields of the Change Datatype

The change datatype has the following fields as shown in the following table:

Name	Type	Description
op	ChangeOp	The change operation; edit, ins, del, repl, or move.
target	xsd:string	The target of the change specified in directory path format, relative to the object. Even though the target is always present, it can be an empty string if the change is relative to the root of an object, such as a VirtualMachine or Host object.
editPos	xsd:int	(Optional; not supported in this release) The edit cursor position for an edit operation.
deleted	xsd:int	(Optional) The number of characters or composites being deleted in this change, if any.

Name	Type	Description
inserted	xsd:int	(Optional) The number of characters or composites being inserted in this change, if any.
val	xsd:anyType	(Optional) The new leaf or composite value for the change.

op and target are the only two fields that are always present in the XML element for a change. A client must interpret a missing editPos, deleted, or inserted field as if the corresponding field is present with a value of 0 (zero). The val field is present if the inserted field is not 0 (zero).

ChangeConflictFaultInfo

ChangeConflictFaultInfo provides information about the fault that occurs when there is a conflict in a change operation.

Name	Type	Description
handle	xsd:string	Actual handle (or vHandle) of the object on which the change was attempted that resulted in the change conflict.
target	xsd:string	The target of the change, in directory path format, that caused the conflict. The target path is always relative to the root of the object.

ChangeOp

ChangeOp describes the kind of change. There are fields marked "Not Applicable" for this datatype. For these "Not Applicable" fields, the Web service may or may not send these fields. The client must simply ignore these "Not Applicable" fields when processing the corresponding ChangeOp. This behavior may change from one Web service version to another.

Name	Type	Description
val	xsd:string	Takes one of the following values: • del - Delete change operation. • edit - Edit change operation. • ins - Insert change operation. • move - Moves an object from one Container to another. • repl - Replace change operation.

Edit Change Operation

The client may send an edit change operation only when changing leaf values. Similarly the Web service will send the edit change operation only when changing leaf values (Change Targets on page 41). We can further categorize leaf values can be further categorized as string values or non-string values.

String Values — These are values of type xsd:string. The following fields of Change are relevant when string values are edited. The usage of these fields is identical for the GetUpdates and PutUpdates operations.

Note: The Edit change operation is *not* supported on string values in this release. Clients cannot perform partial updates on a string. If the Edit change operation is called on a string, the result is similar to a Replace change operation, where the entire string is replaced.

- `target` — Interior node whose value is being edited. An example is `Item["xxx"]/name` when the value for the interior 'name' node is being edited for an Item whose handle is `"xxx"`.
- `editPos` — (Not supported for this release) For deletions, this specifies the starting position of the substring being deleted. For insertions, this is the character position before which a substring is inserted. If this field is NULL, it must be interpreted as `0` (zero).
- `deleted` — Number of characters being deleted starting at `editPos`. If this field is NULL, it must be interpreted as `0` (zero).
- `inserted` — Number of characters being inserted before the editPos character. If this field is NULL, it must be interpreted as `0` (zero).
- `val` — this contains the substring to be inserted if the `inserted` field is non-zero.

If an entire string is being replaced by using the edit operation, then `editPos` is `0` (zero), `deleted` is the length of the old string, `inserted` is the length of the new string, and `val` contains the new string. This is a common case where the edit operation is used to replace an entire string.

Non-string values — These are leaf values not of type xsd:string. The edit operation for such values always replaces the existing value with a new value. The client can ignore the `editPos`, `inserted`, and `deleted` fields of Change. Similarly, when doing a PutUpdates operation, the Web service ignores these fields for non-string values. The usage of Change fields for edit for non-string values is:

- `target` — Interior node whose value is being edited. An example is `hardware/cpu/controls/shares` where the value for the shares node is being replaced for a virtual machine.
- `editPos` — Not Applicable.

- `deleted` — Not Applicable.
- `inserted` — Not Applicable.
- `val` — Contains the new value that replaces the old value on the target.

Insert, Delete, or Replace (ins, del, or repl) Change Operations

Clients can send these change operations for composites, for both arrays and non-arrays.

Note: The Replace change operation is not allowed on array scalar values. The client must instead use an Edit operation.

The usage of Change fields for `ins`, `del`, or `repl` is:

- `target` — Composite that is being inserted, deleted, or replaced. An example is `hardware/net/adapter["#_nic0"]` where the network adapter with key `"#_nic0"` is either being inserted, deleted, or replaced.
- `editPos` — Not Applicable.
- `deleted` — Number of composite nodes being deleted.
- `inserted` — Number of composite nodes being inserted.
- `val` — Contains the new composite or set of composites being inserted or replaced. When the change operation is delete, this field is NULL.

As an example, a single replace change can replace one `#_nic0` adapter with two adapters `#_nic1` and `#_nic2`. In this case, the changeOp is `repl`, the target path is `hardware/net/adapter["#_nic0"]`, `deleted` is `1`, `inserted` is `2`, and `val` contains an array of the two new adapters `#_nic1` and `#_nic2`. Note that the keys are assigned by the Web service.

When the array is an indexed array, the target contains the index of the array element before or at which the operation occurs. For example, if `target` is `vm[1]`, `deleted` is `0`, `inserted` is `2`, and `val` contains two new `vm` array elements, then the new array elements are inserted before the existing `vm[1]`. The first of these new elements becomes `vm[1]`, the second new element becomes `vm[2]` and the old `vm[1]` is now `vm[3]`. In this same example, if `deleted` is also set to `1`, then the old `vm[1]` gets deleted and the two new elements are inserted in its place as `vm[1]` and `vm[2]`.

Move Change Operation

The Web service uses this change operation to indicate a move of an object from one Container to another. This operation can never be used from a client to the Web service in a PutUpdates call. To move objects, clients must instead use the Rename operation.

When an object is moved from a source Container to a destination Container, a GetUpdates call on both the source and destination Container handles gets a Change with a 'move' change operation. The source Container will have `deleted` set to `1`, indicating that the object has moved out of the

source Container. The destination Container will also see a `move` operation with `inserted` set to `1`, to indicate that the object has moved into that Container.

The usage of Change fields for move is:

`target` — Composite which is being moved. An example is `Item["xxx"]` where `Item` whose handle is `"xxx"` is being moved.

`deleted` — Set to `1` in the update to the source Container.

`inserted` — Set to `1` in the update to the destination Container.

`val` — Non-NULL only in the update to the destination Container. val contains the body of the object that has moved.

ChangeReq

ChangeReq describes the change requisition for an object.

Name	Type	Description
handle	xsd:string	Refers to either a handle or a vHandle during a PutUpdates operation. A client uses a handle instead of a vHandle when it wants to update an object value, regardless of any other updates that may have concurrently occurred to that value.
change	Change[ ]	Describes a modification to an object. It specifies the kind of change (edit, insert, delete, replace, or move), where the change took place, and the new value (if applicable).

ChangeReqList

ChangeReqList provides a list of change requisitions.

Name	Type	Description
req	ChangeReq[ ]	List of change requisitions (ChangeReq). A ChangeReq comprises a handle (or a vHandle) and an array of Change types as described in Change on page 41.

ConfigLimits

ConfigLimits provides the minimum and maximum limits for devices and supported guest operating systems.

Name	Type	Description
maxAudio	xsd:int	Maximum limit on Audio devices supported by the host. (Maximum number is 1.)
maxFloppy	xsd:int	Maximum limit on the Floppy drives supported by the host. (Maximum number is 2).
maxNet	xsd:int	Maximum limit on the Network Devices supported by the host. (Maximum number is 3).
maxParallel	xsd:int	Maximum limit on the Parallel Devices supported by the host. (Maximum number is 3).
maxSerial	xsd:int	Maximum limit on the Serial devices supported by the host.(Maximum number is 4).
maxUSBController	xsd:int	Maximum USB controller devices supported by the host. (Maximum number is 1.)
ideLimits	ControllerConfigLimits	Limits for IDE Controller devices on the host.
scsiLimits	ControllerConfigLimits	Limits for the SCSI Controller devices on the host.
guestOS	GuestOSInfo[ ]	(Optional) Set of GuestOSInfo structures, that provides the limits information for each guest operating system supported by the host.
vmLimit	VMMemoryLimits[ ]	(Optional) Keyed array that provides the amount of memory available for creating virtual machines on the host for a given number of CPUs on the virtual machine. The first element of this array provides the memory limit for one CPU virtual machines while the second element provides the memory limit for two CPU virtual machines.

Container

A Container is an object that contains an array of Items. It is used to represent VMware VirtualCenter Server Farm Groups, and appears as the root of the `/vcenter` hierarchy, as well as in nodes in this hierarchy.

A Container is also used as a general container, and therefore appears as the root of the `/host` and the `/vm` hierarchy. See GetContents on page 120 for a complete list of nodes where a Container may be present.

Name	Type	Description
item	Item[ ]	(Optional) Array of Items, that can be of type Container, VirtualMachineGroup, Farm, and so on. See GetContents on page 120 for the complete list of Item types.

ControllerConfigLimits

ControllerConfigLimits provides the maximum number of supported controllers supported and the maximum number of devices per controllers on the host.

Name	Type	Description
maxControllers	xsd:int	For IDE or SCSI Controllers, the maximum limit on controller devices supported by the host. The maximum number of IDE controller devices is 2 while the maximum number of SCSI controller devices is 1.
maxDevicesPerController	xsd:int	Maximum number of devices supported per Controller, that are available on the host. The maximum number of IDE devices per Controller is 2 while the maximum number of SCSI devices per controller is 7.

CustomPropertyList

CustomPropertyList is a list of custom properties.

Name	Type	Description
customProperty	Property	(Optional) A custom property is a user-defined property created through VirtualCenter. The property applies to all hosts and virtual machines. It can be viewed through the VirtualCenter Webservice using GetContents. CustomPropertyDef is a global list of all properties currently defined in the VirtualCenter.

DailyTask

DailyTask describes a task that runs once every day or once every few days, depending on the interval you specify.

Name	Type	Description
hours	xsd:int	The task is scheduled at this hour of the day. For example, "20" specifies that this task should run at 8 p.m.
minutes	xsd:int	Specifies the minutes of the hour when the task is scheduled. For example, "12" specifies that this task should run 12 minutes after the hour.
interval	xsd:int	Specifies the number of days (interval) between scheduled tasks. For example, "2" specifies that this task should run every two days.

DatastoreInfo

A datastore is a repository of files.

Name	Type	Description
key	xsd:string	The name of the datastore as a "datastore file path" specifying just a datastore.
description	xsd:string	(Optional) User-visible description of the datastore.
host	xsd:string	(Optional) Array of handles to hosts that can use the datastore.
location	xsd:string	URL of the root directory of the datastore.
capacityMB	xsd:int	Capacity of the datastore, in MB.
freeSpaceMB	xsd:int	Amount of free space currently available on the datastore, in MB.

DatastoreInfoList

DatastoreInfoList describes a list of DatastoreInfo.

Name	Type	Description
datastore	DatastoreInfo[ ]	(Optional) List of DatastoreInfo.

DiskAdapterType

DiskAdapterType describes the type of SCSI adapter.

Name	Type	Description
val	xsd:string	Takes one of the following values: • busLogic • lsiLogic

DiskControllerType

DiskControllerType describes the type of disk controller.

Name	Type	Description
val	xsd:string	Takes one of the following values: • scsi • ide • floppy • cdrom

DiskControls (ESX Server Only)

DiskControls describes resource management for all the virtual disks in a single virtual machine.

Name	Type	Description	Modifiable by PutUpdates
shares	xsd:string	Per-virtual-disk shares allocation, number in [1, 100000], or special named value such as “low”, “normal”, or “high”.	Yes

DiskDeviceAccessType

DiskDeviceAccessType describes the allocation of the disk drive.

Name	Type	Description
val	xsd:string	Takes one of the following values: • partition • wholeDisk

DiskDeviceInfo

DiskDeviceInfo provides information about the disk device if the DiskType is "device". If the client is creating a disk with raw disk mappings, then only the partitionType field is used. For more information, see DiskType on page 54.

Note: The DiskDeviceInfo datatype is not implemented for CD-ROM and floppy devices.

Name	Type	Description	Modifiable by PutUpdates
accessType	DiskDeviceAccessType	Specifies whether the disk device is a "partition" or a "wholeDisk". Select "wholeDisk" if you are creating a virtual machine with raw disk mapping. In this current release, the accessType stays at the default value of "wholeDisk". Setting the accessType to "partition" is ignored.	No
start	xsd:long	(Optional and currently not used) If the disk device is a partition, then you can specify the start of the partition.	No
sizeMb	xsd:int	(Optional and currently not used) If the disk device is a partition, then you can specify the size of the partition, in MB.	No
partitionType	xsd:string	(Optional) Select either "physical" or "virtual" if you are creating a virtual machine with raw disk mapping. • physical — Guest operating system accesses the disk directly. • virtual — Virtual machine can use the different VMware disk modes (undoable, persistent, and so on) and other advanced functionality.	Yes

DiskFileInfo

DiskFileInfo provides information about the disk file if the DiskType is "file". For more information, see DiskType on page 54.

Note: The DiskFileInfo datatype is not implemented for CD-ROM and floppy devices.

Name	Type	Description	Modifiable by PutUpdates
sizeMb	xsd:int	Specifies the size of the virtual disk file, in MB. **Note:** After creating a virtual disk, you cannot change the sizeMb with a PutUpdates call. If you want to change the sizeMb, then delete the existing virtual disk, and create a new one with a different sizeMb.	Yes, when creating the virtual disk.
flat	xsd:boolean	(Optional) Specifies whether or not the virtual disk file is "flat" (disk storage is preallocated). The default value is false. • ESX Server — This field should be unspecified (the Java default is to set it to "null"). • GSX Server — If set to true, then all disk space is preallocated. If set to false, then disk space is allocated only as needed. **Note:** After creating a virtual disk, you cannot change the value for "flat" with a PutUpdates call. If you want to change "flat", then delete the existing virtual disk, and create a new one with a different value for "flat".	Yes, when creating the virtual disk.

Name	Type	Description	Modifiable by PutUpdates
monolithic	xsd:boolean	(Optional) Specifies whether or not the virtual disk file is "monolithic" (one large virtual disk file or split into 2GB chunks). The default value is false. • ESX Server — This field should be unspecified (the Java default is to set it to "null"). • GSX Server — Set to false or true. If false, then the files are allocated in 2GB chunks. **Note:** After creating a virtual disk, you cannot change the value for "monolithic" with a PutUpdates call. If you want to change "monolithic", then delete the existing virtual disk, and create a new one with a different value for "monolithic".	Yes, when creating the virtual disk.

DiskKind

DiskKind describes the type of virtual disk.

Name	Type	Description
val	xsd:string	Takes one of the following values. Select "device" for a virtual machine with raw disk mapping to a raw disk volume. • device • file

DiskPerf

DiskPerf provides performance data for virtual or physical disks.

Name	Type	Description	Applies to
numRd	xsd:long	(Optional) Number of blocks read in an interval.	ESX Server only
bytesRd	xsd:long	(Optional) Number of bytes read in an interval.	ESX Server GSX Server
numWr	xsd:long	(Optional) Number of blocks written in an interval.	ESX Server only
bytesWr	xsd:long	(Optional) Number of bytes written in an interval.	ESX Server GSX Server

DiskType

DiskType describes the structure of the virtual disk that you are adding to a virtual machine.

Name	Type	Description	Modifiable by PutUpdates
diskKind	DiskKind	The diskKind is either a "device" or a "file".	Yes
diskDeviceInfo	DiskDeviceInfo	(Optional) If diskKind is a device, then you can specify additional information about the device. See DiskDeviceInfo on page 51.	Yes
diskFileInfo	DiskFileInfo	(Optional) If diskKind is a file, then you can specify additional information about the file. See DiskFileInfo on page 52.	Yes

Event

Event describes an individual event. However, each event may affect more than one object. For example, an operation on a virtual machine usually sets the virtual machine, its host, and its Farm.

Name	Type	Description
key	xsd:string	Internal name for the event object.
decl	xsd:string	Declaration corresponding to this event.
arg	EventArgInfo[ ]	(Optional) Parameters for this event that are applied to the declaration.
parent	xsd:string	(Optional) If applicable, the event log index corresponding to the parent of this event.
timeStamp	xsd:dateTime	Time the event was generated. This timestamp element is not adjusted for daylight savings time (DST).
userDesc	xsd:string	(Optional) User description of the event.
farm	EventArgInfo	(Optional) Information about the Farm that "owns" this event.
host	EventArgInfo	(Optional) Information about the host that "owns" this event.
vm	EventArgInfo	(Optional) Information about the virtual machine that "owns" this event.
schedule	xsd:string	(Optional) Event schedule.

EventArgInfo

EventArgInfo describes the arguments for an event.

Name	Type	Description
type	xsd:string	Argument type.
val	xsd:string	Argument value; either a string, if the type is a string, or a handle to an object, if the type is a virtual machine or host.
userDesc	xsd:string	(Optional) User-readable name of the argument value, if it is an object.

EventCollection

EventCollection specifies all the available events.

Name	Type	Description
event	Event[ ]	(Optional) Event log list

EventCollector

EventCollector is a view that displays subsets of all events, based on a filter.

Name	Type	Description
filter	EventFilter	Filter used to determine the events that are collected.
events	EventCollection	(Optional) Actual events that are collected. This field is omitted when creating the EventCollector datatype.

EventDecl

EventDecl describes an individual event declaration; that is, a declaration of a particular event type.

Name	Type	Description
Key	xsd:string	ID of the event declaration.

Name	Type	Description
kind	EventKind	Type of event. Takes one of the following values: • alert • error • warning • info • user
argType	EventArgInfo[ ]	(Optional) The argument type for the event. For more information, see EventArgInfo on page 55.
msgFmt	xsd:string[ ]	(Optional) The format of the event message.

EventDeclList

EventDeclList describes all event declarations.

Name	Type	Description
decl	EventDecl[ ]	(Optional) Event declaration list.

EventFilter

EventFilter describes the filtering attributes used to determine the events that are collected.

Name	Type	Description
parent	xsd:string	(Optional) Collects only events with this parent event.
schedule	xsd:string	(Optional) Collects events created by this schedule.
kind	EventKind	Collects events of this kind.
startTime	xsd:dateTime	Collects events after this start time.
endTime	xsd:dateTime	Collects events before this end time.
farm	xsd:string	(Optional) Collects events owned by this Farm.
host	xsd:string	(Optional) Collects events owned by this host.
vm	xsd:string	(Optional) Collects events owned by this virtual machine.
declId	xsd:string	(Optional) Collects events with only this event declaration ID.
totalEvents	xsd:int	(Optional) Collects this number of events, giving priority to more recent events.

EventKind

EventKind describes the type of event.

Name	Type	Description
val	xsd:string	Takes one of the following values: • alert • error • warning • info • user

Farm

A Farm contains an array of Items; each Item may be one of the following:

- Hosts
- Virtual machines
- Virtual Machine Groups

Name	Type	Description
item	Item[ ]	(Optional) Array of Items that can be of type VirtualMachineGroup, Host and VirtualMachine.

FaultInfo

FaultInfo describes an error detected by the Web service during the processing of a request.

Name	Type	Description
kind	FaultKind	Specifies the fault that occurred.
code	xsd:int	(Optional) Integer, which gives the error code that is classified in a manner similar to HTTP error codes.

Name	Type	Description
info	xsd:string	(Optional) This field contains additional details for certain kinds of faults: • AlreadyExists, BadName, NotDirectory, NotFound, or PermissionDenied — Contains a string with the remainder of a path where the exception occurred. • BadRequest — Contains text from the input for the error. • BadState — Contains details that describe the state of the virtual machine when the exception occurred. • ChangeConflict — Contains details about the target expression that failed. • NotApplicable — Contains the handle that is the incorrect type (for example, StartVM on a host). • OperationFailed — Contains an error message that is returned from the server.
val	xsd:anyType	(Optional) This field contains structured data for certain kinds of faults: • HandleFault — Contains a HandleFaultList with at least one HandleFaultInfo containing specific data.

FaultKind

FaultKind describes the kind of fault encountered during the processing of a request.

Name	Type	Description
FaultKind	xsd:string	Takes one of the following values. The definitions for each of these faults follow this table. • AlreadyExists • BadName • BadParameter • BadRequest • BadState • ChangeConflict • HandleFault • Interrupted • NoResources • NotApplicable • NotDirectory • NotFound • OperationFailed • PermissionDenied

- AlreadyExists — The specified target already exists.
- BadName — The supplied name parameter contains one or more of the reserved characters in the path segments ("?;=" is the list of reserved characters).
- BadRequest — The SOAP or XML request is not well formed or the parameters are not valid.
- BadState — The state of the object (virtual machine, host, and so on) does not permit the specified operation.
- ChangeConflict — During a PutUpdates operation, the Web service is unable to accept the changes that the client sent over, because the vHandle specified by the client does not match the current vHandle in the Web service, and the changes conflict with another change already made on the Web service. The client should run the GetContents operation on the object again, and then retry the change.
- HandleFault — One or more of the handles passed as a parameter has encountered errors. The details of the error are described in the HandleFaultList that is available in the FaultInfo.val document.

- Interrupted — During a blocking GetUpdates operation, one or more of the specified vHandles has become obsolete.
- NoResources — Internal Web service error due to lack of memory.
- NotApplicable — The specified operation could not be performed on the object.
- NotDirectory — While resolving the path, one of the intermediate path elements is not a directory.
- NotFound — The specified object could not be located.
- OperationFailed — The operation could not be successfully completed by the server. Clients may also see this FaultKind if invalid parameters are passed to an operation.
- PermissionDenied — Login authentication failed or permission for the specified path was not available for ResolvePath.

Note: Clients see either the BadRequest or OperationFailed FaultKind when invalid parameters are passed to an operation.

GenericDeviceInfo

GenericDeviceInfo provides information for non-disk and non-network physical or virtual devices, including: video, usb, scsi, serial, parallel, and audio. However, the mouse used with GSX Server is not handled by this datatype.

Note: The GenericDeviceInfo datatype is read-only and may not be modified by a PutUpdates operation.

Name	Type	Description
key	xsd:string	Key for the device.
name	xsd:string	Logical name of the device.
type	xsd:string	Type of device. (For GSX Server only, also includes a sound card or USB controller.
removable	RemovableDeviceInfo	(Optional) Describes parameters for devices that may be disconnected from the virtual machine.
property	Property	(Optional) All properties of the device. The list of properties vary, and are dependent on the device.

GenericDevicePerf (ESX Server Only)

GenericDevicePerf describes performance for non-disk and non-network physical or virtual devices.

Name	Type	Description
bytesRd	xsd:long	(Optional) Number of bytes read in an interval.
bytesWr	xsd:long	(Optional) Number of bytes written in an interval.

Guest Customization Datatypes

Through VirtualCenter or when calling the CloneVM operation, the client has an option to customize the identity and network settings of a guest operating system so that the new virtual machine can begin work immediately in the target environment. See CloneVM on page 187.

We call the data object that contains this information, the Customization Specification. This section describes each of the guest customization datatypes and the various fields it contains.

- Autoprep on page 61
- Options (Windows Only) on page 62
- Sysprep (Windows Only) on page 62
- Linux-Global (Linux Only) on page 66
- Adapters on page 67

Autoprep

The top level datatype of the guest customization specification is usually named autoprep, although this is not a requirement. This datatype comprises attributes that identify the document type and version. Each attribute describes different aspects of guest customization.

Name	Type	Description
options	Options	(Windows only.) Describes the entire customization of Windows guest operating systems
sysprep	Sysprep	(Windows only.) Describes customization operations that are implemented by Microsoft's sysprep automated deployment tool
linux-global	Linux-global	(Linux only.) Describes customization options for a Linux guest operating system.
adapters	Adapters	(Optional) Describes customization options for network parameters.

Options (Windows Only)

This datatype comprises a set of options that affect the entire customization of Windows guest operating systems.

Name	Type	Description
changeSID	xsd:boolean	(Optional) Takes a value of true or false. The virtual machine's access to network resources and its membership in a domain are based on the machine's SID. (The SID, or Security ID, is a unique ID for an Windows machine.) If the changeSID field has a value of true, then the machine gets a new, randomly generated SID from the sysprep tool as part of the guest customization.
deleteAccounts	xsd:boolean	(Optional) This field is useful when cloning a virtual machine from a template or an existing virtual machine. It takes a value of true or false. If true, then the customization process removes all user accounts from the target guest operating system, including the Administrator account. During customization, a new Administrator account is automatically created with a blank password, which can then be customized using the AdminPassword field. See GuiUnattended (Windows Only) on page 63. If this option is set to false, then the Administrator password for the source template (or existing virtual machine) must be blank, or the customization process will be unable to set the Administrator password. If an Administrator account with a non-blank password exists on the virtual machine, then the AdminPassword tag has no effect and the virtual machine retains the Administrator password from the source template (or existing virtual machine). **Note:** Setting this option to true causes the removal of the SAM registry hive file from the target guest operating system. In some situations, this modification may leave the guest operating system in an inconsistent security state. We recommend that you *not* use this option when customizing Windows XP or Windows 2003 guest operating systems.

Sysprep (Windows Only)

This datatype comprises customization operations that are implemented by Microsoft's sysprep automated deployment tool and are only for Windows guest operating systems. This tool makes it possible to modify the virtual machine's identity, certain regional settings, and to enter a serial number for a new installation. Extensive documentation for sysprep can be found at *www.microsoft.com/technet/prodtechnol/Windows2000Pro/deploy/depopt/sysprep.mspx*.

The parameters and their structure in the sysprep section of the customization specification are similar to those of the `sysprep.inf` answer file used to drive the sysprep tool.

If you plan to customize a Windows guest operating system, then you must first install the Microsoft Sysprep tools package on the VirtualCenter server machine. Follow the procedure in the section titled "Preparing for Guest Customization" in the *VMware VirtualCenter User's Manual*.

Note: The sysprep datatype is relevant only to Windows guest operating systems and is generally not included when customizing Linux guest operating systems.

Name	Type	Description
GuiUnattended	GuiUnattended	Contains entries for preparing the graphical user interface (GUI) for unattended installation.
UserData	UserData	Contains entries for user or virtual machine information.
GuiRunOnce	GuiRunOnce	(Optional) Contains commands that run the first time a user logs onto the virtual machine.
Identification	Identification	Contains entries for specifying the network identification of a computer.
LicenseFilePrintData	LicenseFilePrintData	(Optional) Contains entries for installing licensing information for Windows server guest operating systems.

GuiUnattended (Windows Only) — GuiUnattended is part of the sysprep datatype. It contains entries for preparing the graphical user interface (GUI) for unattended installation.

Name		Description
AdminPassword	Password	Sets the Administrator account password for the new virtual machine. **Note:** The customization process only assigns the new Administrator password if the current password is blank. If the Administrator password is not blank, then the AdminPassword field has no effect. There are two methods for changing the Administrator password: • Check and make sure that the Administrator password in the template or source (existing) virtual machine is blank. • Use the deleteAccounts option. See Options (Windows Only) on page 62. This option removes all user accounts, including the Administrator account. When the new virtual machine boots, the guest operating system automatically creates a new Administrator account, with a blank password. If you want the new guest operating system's Administrator password to be in plain text, then be sure to set the plaintext attribute (of type xsd:boolean) for AdminPassword to true. If you want the new guest operating system's Administrator password to be blank, then specify an asterisk for the value; for example: `AdminPassword="*";`
TimeZone	xsd:string	Integer that specifies the time zone of the new virtual machine. See Valid TimeZone Indexes on page 213.
AutoLogon	xsd:boolean	(Optional) Can take a value of true or false. If this value is true, then the Windows guest operating system automatically logs onto the Administrator account using the password specified in the AdminPassword tag. If this value is false, then the logon window appears.
AutoLogonCount	xsd:integer	(Optional) Applies only if AutoLogon is set to "true." Specifies the number of times that the virtual machine automatically logs on with the specified Administrator account and password.

UserData (Windows Only) — UserData is part of the sysprep datatype. It contains entries for user or virtual machine information.

Name	Type	Description
FullName	xsd:string	Specifies the user's full name.
OrgName	xsd:string	Specifies the user's organization name.

Name	Type	Description
ComputerName	xsd:string	Specifies the host name for a Linux guest operating system or the computer (network) name for a Windows guest operating system. Specify an asterisk `*` to get a name randomly generated based on the OrgName.
ProductID	xsd:string	(Sometimes required for completely unattended customization.) Specifies the product key for each unique installation of Microsoft Windows. If the template was created using a retail CD, this field is required to automate the setup fully. Otherwise sysprep's mini-setup stops as the guest operating system boots and requests a CD key.

GuiRunOnce (Windows Only) — GuiRunOnce is part of the sysprep datatype. It contains commands that run the first time a user logs onto the virtual machine after the final (GUI-mode) stage of setup completes.Commands called in GuiRunOnce run in the order they are listed, and each command must finish before the next command can run.

Name	Type	Description
Command	xsd:string	(Optional) Each Command field lists a command that runs the first time a user logs on after the new virtual machine is created. At the end of sysprep's mini-setup, these commands are scheduled using the `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce` registry key.

Identification (Windows Only) — Identification is part of the sysprep datatype. It contains entries for specifying the network identification of a computer. If these entries are not present, setup adds the computer to the default workgroup. If there is not enough information for this entry, then setup prompts the user to provide this information.

Note: Either the JoinDomain or JoinWorkgroup fields must be specified.

Name	Type	Description
JoinWorkgroup	xsd:string	(Required if joining a workgroup.) Specifies the name of the Windows workgroup in which the virtual machine participates. If the virtual machine is joining a domain, then see JoinDomain below.
JoinDomain	xsd:string	(Required if joining a domain.) Specifies the name of the Windows domain in which the virtual machine participates. If the virtual machine is joining a workgroup, then see JoinWorkgroup above. If the virtual machine is joining a domain, then the DomainAdmin and the DomainAdminPassword fields must be specified as well.

Name	Type	Description
DomainAdmin	xsd:string	(Required if joining a domain.) Specifies the name of a Windows domain administrator who can add virtual machines to a domain.
DomainAdminPassword	Password	(Required if joining a domain.) Specifies the password for the Windows domain administrator account. In order to enter the password in plain text, then be sure to set the plaintext attribute (of type xsd:boolean) for DomainAdminPassword to true.

LicenseFilePrintData (Windows Server Systems Only) — LicenseFilePrintData is part of the sysprep datatype. It contains entries for installing licensing information for Windows server guest operating systems.

Name	Type	Description
AutoMode	xsd:string	(Optional) Takes a value of either PerServer or PerSeat. Determines whether Windows 2000 Server or Windows 2003 (all versions) are to be installed in a per-seat or a per-server license mode. If AutoMode = PerServer, then the AutoUsers key must also be specified.
AutoUsers	xsd:integer	(Optional) This field is valid only if AutoMode = PerServer. The integer value indicates the number of client licenses purchased for the server that is being installed. The specified number must be greater than five, otherwise unattended Setup mode fails and the user is prompted to pick a value through the user interface.

Linux-Global (Linux Only)

This datatype contains fields that are relevant to a customization of a Linux guest operating system. It provides basic network parameters that pertain to the virtual machine as whole, and not on a per-NIC basis. Linux-global also includes the host name for the virtual machine and its domain name.

If you plan to customize a Linux guest operating system, then you must first install the VMware Open Source components on the VirtualCenter server machine. Click on the download link for Open Sources at *www.vmware.com/download* and follow the procedure in the section titled "Preparing for Guest Customization" in the *VMware VirtualCenter User's Manual*.

Note: The linux-global datatype is relevant only to Linux guests operating systems and is generally not included when customizing Windows guest operating systems.

Name	Type	Description
ComputerName	xsd:string	Name of the host for the new virtual machine.

Name	Type	Description
Domain	xsd:string	Name of the domain, to which the virtual machine belongs.
Domain/DNS/ PrimaryDNS	xsd:string	IP addresses for primary Domain Name Server (DNS) name lookup server. At minimum, the IP address for the primary DNS server is required.
Domain/DNS/ SecondaryDNS	xsd:string	(Optional) IP address for secondary DNS name lookup server.
Domain/DNS/ TertiaryDNS	xsd:string	(Optional) IP addresses for tertiary DNS name lookup server.
Domain/DNS/ DNSSearchPaths/ DNSSearchPath	xsd:string	Domain suffixes to append during name lookup. At minimum, one suffix is required for use during name lookup.

Adapters

This datatype contains fields that customize network parameters to a specific virtual network adapter. These settings are generally included for both Windows and Linux guest operating systems.

Note: There is one adapter section for each virtual NIC on the newly deployed virtual machine. That is, each of these settings is on a per-NIC basis.

Name	Type	Description
MACAddress	xsd:string	Specifies a symbolic name for each NIC that represents the interface's MAC address. When a new virtual machine is deployed, VirtualCenter assigns a new MAC address to each interface. At this point, the new MAC address is mapped to the symbolic name in the customization specification, and all the network parameters are applied to the NIC with that MAC address.
UseDHCP	xsd:boolean	Takes a value of true or false. If this value is true, then the newly deployed virtual machine's guest operating system retrieves its network settings from a Dynamic Host Configuration Protocol (DHCP server) and any static settings included in the adapter tag are ignored. If this value is false, then any settings that are required by the adapter must be included in the customization specification.
NetBios	NetBIOSEnum	(Optional) Specifies that the virtual machine should query the DHCP server to decide whether to enable NetBios over the TCP/IP protocol. Takes one of the following values: • EnableNetBIOSViaDhcp • EnableNetBIOS • DisableNetBIOS

Name	Type	Description
IPAddress	IPAddress	The network interface's static IP address.
SubnetMask	IPAddress	The network interface's static subnet mask.
Gateways/Gateway	Gateway	(At least one Gateway per-NIC.) Each Gateway field specifies an Internet gateway available to the network interface. Windows customization requires that a CostMetric attribute be attached to each Gateway. This attribute specifies the order in which the gateways are used. Gateways with a lower CostMetric value are used first. Because Linux supports only a single Gateway, the CostMetric attribute is ignored.
DNSDomain	xsd:string	(Windows only, at least one Gateway per-NIC.) Specifies the DNS suffix for this network adapter. In Linux, this information is specified on a machine-wide basis by using the linux-global/Domain field.
DNSServers/ DNSServer	IPAddress	(Optional) Specifies a list of domain name servers for name lookup. This per-NIC setting is a counterpart to the machine-wide settings specified in the linux-global/Domain/DNS/PrimaryDNS, SecondaryDNS, and TertiaryDNS fields.
DNSFromDHCP	xsd:boolean	(Optional field, SuSE Linux guest operating systems only.) Takes a value of true or "false." SuSE Linux guest operating systems allow a fine-grained selection of settings that are to be retrieved from DHCP. If true, then the customization process configures a SuSE guest operating system to retrieve its DNS server settings from DHCP.
PrimaryWINS	IPAddress	(Optional) Primary Windows Internet Name Service (WINS server) for the adapter.
SecondaryWINS	IPAddress	(Optional) Secondary WINS server for the adapter.
DNSSuffixes/DNSSuffix	xsd:string	(Optional) Suffixes that are appended to a network name when doing DNS lookup.

GuestInfo

GuestInfo provides information about the guest operating system on the virtual machine. This information is available through GuestInfo only when VMware Tools are installed and running on the virtual machine.

Name	Type	Description
hostname	xsd:string	Host name of the guest operating system.
ipAddress	xsd:string	Primary IP address assigned to the guest operating system.

Name	Type	Description
net	GuestNetworkAdapter[ ]	(Optional) An array of network adapters configured in the guest operating system. Not used in this current release.
toolsStatus	VirtualMachineToolsStatus	Current status of VMware Tools in the guest operating system.
toolsMounted	xsd:boolean	Boolean indicating whether or not tools are mounted in the guest operating system.
heartbeatSampleInterval	xsd:int	Sample interval for heartbeats.
heartbeatsExpected	xsd:int	Number of heartbeats that are expected in the interval specified by the heartbeatSampleInterval.
heartbeatsReceived	xsd:int	Number of heartbeats received in the heartbeatSampleInterval.
config	Property[ ]	(Optional) Other properties, if any, about the guest operating system.

GuestNetworkAdapter

GuestNetworkAdapter provides information about each network adapter configured in the guest operating system. This is not populated for GuestInfo in the VirtualMachine datatype.

Name	Type	Description
key	xsd:string	MAC address of the network adapter.
network	xsd:string	Name of the network connected to this adapter.
ipAddress	xsd:string[ ]	(Optional) IP address of the adapter.

GuestOSInfo

GuestOSInfo is a set of GuestOSInfo structures, that provides the limits information for each guest operating system supported by the host.

Name	Type	Description
key	xsd:string	A short key for the guest operating system. This is used as the key element in a keyed array of GuestOSInfo; for example, `win2000Pro`.
label	xsd:string	A descriptive name for the guest operating system; for example, Windows 2000 Professional.

Name	Type	Description
supported	xsd:boolean	Flag indicating whether or not the guest operating system is supported by the host.
maxCpu	xsd:int	Maximum number of CPUs used by this guest operating system.
defaultDiskSize	xsd:long	Default size of the disk that can be created in a virtual machine with this guest operating system.
defaultColorDepth	xsd:int	Default color depth supported by this guest operating system.
defaultDiskControllerType	DiskControllerType	Default disk controller type is one of two values: ide or scsi. **Note:** ESX Server does not support any guest operating systems for which defaultDiskControllerType is set to ide.
defaultSCSIAdapterType	DiskAdapterType	Default SCSI adapter type for this guest operating system: busLogic or lsiLogic.
memory	GuestOSMemoryInfo	Detailed information about the guest operating system memory.

GuestOSMemoryInfo

GuestOSMemoryInfo provides minimum and maximum limits supported by the guest operating system for the host.

Name	Type	Description
minHostMb	xsd:int	Minimum available host memory required by this guest operating system.
minMb	xsd:int	Minimum available memory required to install and run this guest operating system.
maxMb	xsd:int	Maximum memory supported by this guest operating system.
defaultMb	xsd:int	Default Memory size selected when creating a virtual machine for this guest operating system.

HandleFaultInfo

HandleFaultInfo describes the error for one handle (or vHandle).

Name	Type	Description
kind	HandleFaultKind	Specifies error values for the different handle (or vHandle) faults
handle	xsd:string	Actual handle (or vHandle) that caused the HandleFaultKind.
nextVHandle	xsd:string	(Optional) vHandle the client should use in a subsequent call to the GetUpdates operation.

HandleFaultKind

HandleFaultKind specifies error values for the different handle (or vHandle) faults.

Name	Type	Description
HandleFaultKind	xsd:string	Takes one of the following values. The definitions for each of these faults follow this table. • BadHandle • BadVersion • Canceled • DuplicateHandle • ObsoleteHandle • Unauthorized

- BadHandle — Indicates that either the user session has expired or the Web service has restarted. If the Web service has restarted, then all handles on the Web service have been reset. In this case, we recommend that the client should rebuild its state. A Web service restart is a very infrequent occurrence, but a good client program should be able to address this condition.
- BadVersion — Indicates that the handle is valid, but the version number is too old or unknown. The client should do a GetContents operation to get the latest body (XML document) and vHandle of the object.
- DuplicateHandle — There is at least one other handle or vHandle that refers to the same object.

- ObsoleteHandle — Indicates the underlying object has been deleted. The client should remove the vHandle and associated path from its data structures.
- Unauthorized — Indicates that the client does not have the permissions for the specified operation on the handle.

HandleFaultList

HandleFaultList specifies a list of HandleFaultInfo. When an operation encounters an error with one or more handles (or vHandles), the `FaultInfo.kind` indicates "HandleFault". `FaultInfo.val` has a HandleFaultList that is a list of HandleFaultInfo, which provides detailed information for each handle that had an error.

Name	Type	Description
info	HandleFaultInfo [ ]	An array of error information, for each handle that has an error.

Hardware

Hardware specifies processor, memory, and device information for the hardware on a host.

Name	Type	Description
cpu	CPUInfo[ ]	Processor information, one per processor.
memory	MemoryInfo	Memory information.

Host

Host provides information about the host, active virtual machines, performance data, and events.

Name	Type	Description
info	HostInfo	Information about the host.
hardware	Hardware	Host hardware information.
state	HostState	State of the host and last boot time.
vm	xsd:string[ ]	(Optional) List of handles to the virtual machine known to the host.

Name	Type	Description
path	xsd:string	Full path for a VMware VirtualCenter object beginning with `/vcenter`. **Note:** Any unescaped `/` (slash) character used in this path parameter is regarded as a delimiter between path segments. Any `/` (slash) character not intended for use as a delimiter, will be escaped. Similarly, any `%` (percent) character used in this path parameter will be escaped, unless it is used to start an escape sequence.

HostInfo

HostInfo provides information for a host.

Name	Type	Description	Modifiable by PutUpdates
uuid	xsd:string	SMBIOS (system management BIOS) UUID for the host.	No
hostname	xsd:string	Host name. If known, this parameter lists the Fully Qualified Domain Name (FQDN) for the host; for example, `esx1.vmware.com`.	No
port	xsd:int	The host management daemon listens to this port.	No
system	SystemInfo	Host operating system version information.	No
migrationEnabled	xsd:boolean	(Optional) Describes whether or not this host is enabled for migration.	Yes
datastore	xsd:string[ ]	(Optional) Names of all the datastores that are local to this host. Clients can use these datastores as keys into the DataStoreInfoList array in `/datastore`; for example, `[esx1.vmware.com//RegSuite]`.	No
network	NetworkInfo[ ]	(Optional) Network information, including the key and device for the host. Currently, only the key is available; for example, `key = Adapter0 Network`.	No

Name	Type	Description	Modifiable by PutUpdates
configLimits	ConfigLimits	This structure enumerates all the configuration limits for the host, including a list of the supported guest operating systems for the host.	No
customPropertyDef	xsd:string	Handle to the list of all user-defined custom properties.	No
customProperty	Property[ ]	(Optional) Name/value pairs associated with the custom fields defined in VirtualCenter. Clients may update the existing values by using the PutUpdates operation and can add new properties only through the use of VirtualCenter.	Yes
migrationInfo	MigrationInfo	(Optional) Provides information about a virtual machine migration. Not used in this release.	No

HostSpec

HostSpec provides information that is used to connect to a host machine.

Name	Type	Description
port	xsd:int	(Optional) The host management daemon listens to this port.
userName	xsd:string	(Optional) The user name that authenticates the Web service to the host machine.
password	xsd:string	(Optional) The password that authenticates the Web service to the host machine.

HostState

HostState describes the state of the host.

Name	Type	Description
status	xsd:string	Current state of the host. This field takes one of the following values: • enabled • disabled • unknown
detail	xsd:string	Contains additional information regarding the host. Most clients monitor the status field only. This field can take one of the following values: • connected • disconnected • connecting • disconnecting • reconnecting • removing • upgrading
connected	xsd:boolean	This boolean is set to true when the host is in the connected state in VirtualCenter.
bootTime	xsd:dateTime	Last boot time of the host. Not used in this release.
eventCollector	xsd:string	Handle for event collector for this host.

HourlyTask

HourlyTask describes a task that runs once every hour or once every few hours, depending on the interval you specify.

Name	Type	Description
minutes	xsd:int	Specifies the minutes of the hour when the task is scheduled. For example, "12" specifies that this task should run 12 minutes after the hour.

Name	Type	Description
interval	xsd:int	Specifies the number of hours (interval) between scheduled tasks. For example, "2" specifies that this task should run every two hours.

Item

An Item is a special element of the Container, Farm and VirtualMachineGroup objects, that represents a child object within these parent objects.

Name	Type	Description
key	xsd:string	The value of this field is the handle to the object that is identified by the name field.
name	xsd:string	The name of this Item. The path of the parent object may be concatenated with the name of an Item, to form a new path. This new path points to an object whose type is defined by the Item. The new path may be passed in as a parameter to the ResolvePath operation that will return the same handle contained in the key field. **Note:** Any `/` (slash) character used in this name parameter will be escaped. Similarly, any `%` (percent) character used in this name parameter will be escaped, unless it is used to start an escape sequence.
type	xsd:string	Type of the object represented as a string.
perm	Permission[ ]	(Optional) Access rights to the object identified by the name field. Each entry in this array describes access rights for a particular user or group. If there is an entry corresponding to a particular user and a group to which the user belongs, then the union (combination) of the permissions is used by the Web service for access control checks.

KeyedValue

KeyedValue is a key-value pair for dynamic information. It is used to specify parameters for operations in the TaskSchedule datatype. For more information, see TaskSchedule on page 90.

Name	Type	Description
key	xsd:string	Key for data structure.

Name	Type	Description
val	xsd:anyType	Data structure contents.

Level

Level describes the priority and whether resources are preallocated before the virtual machine migration.

Name	Type	Description
val	xsd:string	Takes one of the following values: • low • normal • high

MemoryControls (ESX Server Only)

MemoryControls describes memory resource management for each virtual machine.

Name	Type	Description	Modifiable by PutUpdates
min	xsd:int	(Optional) Guaranteed minimum memory size for a virtual machine, in MB.	Yes
max	xsd:int	(Optional) Maximum memory size for a virtual machine, in MB. Currently this is the same as MemoryInfo:sizeMb.	No
shares	xsd:string	Memory shares allocation, as a number in [0, 100000], or special named value such as “low”, “normal”, or “high”.	Yes
affinity	xsd:string	(Optional) A node affinity set for a virtual machine, specified as a comma-separated list of NUMA node identifiers like “1”. This affinity is only relevant on NUMA systems, and is normally unspecified, allowing for automatic NUMA locality optimizations.	Yes

MemoryInfo

MemoryInfo provides information about the memory in a physical host.

Name	Type	Description
sizeMb	xsd:int	Total allocated memory size (in MB).

MemoryPerf

MemoryPerf describes the host computer's memory status and configuration.

Name	Type	Description
size	xsd:int	(Optional) Total amount of managed memory, in KB.
active	xsd:int	(Optional) Actively used memory, in KB.

MemoryPerf2 (ESX Server Only)

MemoryPerf2 augments the MemoryPerf datatype. It includes 8 additional fields, shown in the following table.

Note: This new performance statistic field will only be available to the client by using the new QueryPerfData2 operation.

Name	Type	Description
size	xsd:int	(Optional) Total amount of managed memory, in KB.
active	xsd:int	(Optional) Actively used memory, in KB.
sizeUnreserved	xsd:int	(Optional) Current amount of unreserved memory, in KB. Used for admission control decisions (for example, to guarantee MemoryControls:min).
swap	xsd:int	(Optional) Total amount of swap space, in KB.
swapUnreserved	xsd:int	(Optional) Current amount of unreserved swap space, in KB. Used for admission control decisions; for example, to guarantee MemoryControls:max.
shared	xsd:int	(Optional) Current amount of shared guest operating system memory, in KB.

Name	Type	Description
sharedCommon	xsd:int	(Optional) Current amount of memory, in KB, consumed for (a single copy of) shared pages. Memory that is reclaimed by page sharing is "shared - sharedCommon".
heap	xsd:int	(Optional) Total amount of system heap memory, in KB.
heapFree	xsd:int	(Optional) Current amount of available system heap memory, in KB.
state	xsd:int	(Optional) Describes the contention for memory and takes one of the following values. The higher the number, then the memory state is more constrained. • 0 — high (lot of memory available) • 1 — soft • 2 — hard • 3 — low (memory is very overcommitted)

MigrationInfo

MigrationInfo provides information about VMotion settings on a host and allows these settings to be changed.

Name	Type	Description	Modifiable by PutUpdates
network	xsd:string	Network name of the virtual switch bound to an outgoing network adapter that is used for VMotion; for example, Adapter0 Network. Use this field to select the host network adapter for VMotion.	Yes (when enabling or disabling migration)
ipAddress	xsd:string	Unique ipAddress for the network setup for VMotion.	Yes (when enabling or disabling migration)
gateway	xsd:string	Gateway address used by the host network adapter. This gateway should be the same across all the hosts that are part of the VMotion network.	Yes (when enabling or disabling migration)
subnet	xsd:string	(Optional) Subnet mask for the VMotion network. (Not currently used.)	Not currently used.

Note: In order to perform the VMotion of virtual machines on a host, the host should have a virtual switch that is bound to an outbound network adapter. This network adapter and virtual switch used for VMotion should be on a separate network from any other network adapter and

virtual switch on the host. You can set up the network adapter and virtual switch for VMotion through VirtualCenter by right-clicking a host and selecting Edit Host Configuration. The VMware Management Interface is displayed in a browser.

MonthlyTask

MonthlyTask describes a task that runs once every month or once every few months, depending on the interval you specify.

Name	Type	Description
hours	xsd:int	The task is scheduled at this hour of the day. For example, "20" specifies that this task should run at 8 p.m.
minutes	xsd:int	Specifies the minutes of the hour when the task is scheduled. For example, "12" specifies that this task should run 12 minutes after the hour.
interval	xsd:int	Specifies the number of months (interval) between scheduled tasks. For example, "2" specifies that this task should run every two months.
day	xsd:int	(Optional) Specifies the day of the month when the task is scheduled. For example, "15" specifies that this task should run on the 15th of the month. **Note:** Be sure to specify a day that actually exists. Otherwise, an error will be generated. For example, if you specify "31" and the month has only 30 days, then an error is generated. **Note:** Specify either "day" or "week". You cannot specify both fields.
week	xsd:int	(Optional) Specifies the week of the month when the task is scheduled. For example, "2" specifies that this task should run in the second week of the month. If you choose "week", then you must also specify the day of the week, as described in the following table row. **Note:** Specify either "day" or "week". You cannot specify both fields.

Name	Type	Description
dayOfWeek	WeekDay	(Optional) If you choose "week", then you must also specify the day of the week. Takes one of the following values: • sunday • monday • tuesday • wednesday • thursday • friday • saturday

NetworkControls (ESX Server Only)

NetworkControls provides information about network traffic shaping.

Name	Type	Description	Modifiable by PutUpdates
shapingEnabled	xsd:boolean	(Optional) Indicates whether or not outbound network traffic should be shaped using the nfshaper traffic shaper with the parameters listed below, in this table.	Yes
averageRate	xsd:int	(Optional) Sustainable long-term average bandwidth allocation, in bits per second (bps).	Yes
peakRate	xsd:int	(Optional) Burstable short-term peak bandwidth allocation, in bps.	Yes
burstSize	xsd:int	(Optional) Maximum burst size, in bytes.	Yes

NetworkInfo

NetworkInfo provides information about a host network interface.

Name	Type	Description
key	xsd:string	Key, that is the logical name of the network interface.
device	xsd:string	(Optional) Host device name for the network interface. Not used in this current release.

NetworkPerf

NetworkCounter provides performance information for a network interface.

Name	Type	Description	Applies to
packetsIn	xsd:long	(Optional) Packets read from network, for each NIC.	ESX Server only
bytesIn	xsd:long	(Optional) Bytes read from network, for each NIC.	ESX Server GSX Server
packetsOut	xsd:long	(Optional) Packets written to network, for each NIC.	ESX Server only
bytesOut	xsd:long	(Optional) Bytes written to network, for each NIC.	ESX Server GSX Server

OneTimeTask

OneTimeTask describes a task that runs only once.

Name	Type	Description
runTime	xsd:dateTime	Specifies the time and date of this task. If the task should run now, or should have previously run, then the runTime is "now".

PerfCollection

PerfCollection provides performance data that is reported by a performance collector.

Name	Type	Description
startTime	xsd:dateTime	Start time of samples that are returned.
source	PerfSource[ ]	(Optional) List of statistics from various sources.

PerfCollector

PerfCollector specifies a collector of performance statistics. The performance collector is a VirtualCenter perf collector (specifies only a sample interval) or a filtered perf collector (derived from and existing VirtualCenter perf collector). For more information on perf collectors, refer to the *Virtual Infrastructure SDK Programming Guide*.

Name	Type	Description
filter	PerfFilter	Specification of the sample interval and the statistics that are collected by the performance collector or the statistics that are filtered from another (parent) performance collector.
stats	PerfCollection	(Optional) Actual stats from specified sources. This field is omitted when creating a PerfCollector.

PerfFilter

PerfFilter describes the filter on a PerfCollector.

Name	Type	Description
interval	xsd:long	Time interval between samples, in seconds.
spec	PerfSourceType[ ]	(Optional) Specification of the performance statistics that will be collected.
name	xsd:string	(Optional) Name for the PerfCollector.
handle	xsd:string	(Optional) Handle of the parent collector. If there is no parent, this is omitted.
samples	xsd:int	(Optional) Number of samples provided by the PerfCollector.
update	xsd:int	(Optional) Number of new samples before generating an update.
totalSamples	xsd:int	(Optional) The total number of the available historical samples. This field is optional.

PerfSample

PerfSample specifies performance statistics for a single sample from a single source.

Name	Type	Description
key	xsd:string	Key of the sample.
interval	xsd:long	Actual sample interval, in seconds.
startTime	xsd:dateTime	(Optional) Start time for this sample. This field is omitted, if it is the same as the end time of the last interval.

Name	Type	Description
stat	PerfStat[ ]	(Optional) List of performance statistics from a source for a particular time period for a single sample.

PerfSource

PerfSource describes performance statistics from a single source.

Name	Type	Description
key	xsd:string	Key for the source of statistics; a handle for the host or virtual machine.
sample	PerfSample[ ]	(Optional) List of samples from the source, ordered by the time of the sample.

PerfSourceType

PerfSourceType describes the specification of performance statistics from various sources.

Name	Type	Description
key	xsd:string	Key for the source of statistics; a handle for the host or virtual machine.
sample	PerfType[ ]	(Optional) List of types of performance statistics that are being collected. Omit this parameter to gather all statistics types.

PerfStat

PerfStat describes a performance statistic for a specified element (for example, a CPU or a network card) from a single source (a host or a virtual machine).

Name	Type	Description
key	xsd:string	Key of this sample.
type	PerfStatType	Type of performance statistic that is reported.
device	xsd:string	Name of the object (for example, a disk) that is reported.
data	xsd:anyType	Statistics for the object (for example, CPUPerf, MemoryPerf, NetworkPerf, and so on).

PerfStatType

PerfStatType describes the type of performance statistic.

Name	Type	Description
val	xsd:string	Takes one of the following values: • cpu - CPU performance data (reported as CPUPerf) • net - network interface performance data (reported as NetworkPerf) • disk - disk performance data (reported as DiskPerf) • floppy - floppy performance data (reported as DiskPerf) • cd - CD performance data (reported as DiskPerf) • device - performance data for other devices connected to the host (reported as GenericDevicePerf) • memory - memory performance data (reported as MemoryPerf) • vcpu - virtual CPU performance data (reported as CPUPerf) • vnet - virtual network interface performance data (reported as NetworkPerf) • vdisk - virtual disk performance data (reported as DiskPerf) • vfloppy - virtual floppy performance data (reported as DiskPerf) • vcd - virtual CD performance data (reported as DiskPerf) • vdevice - performance data for other virtual devices connected to the host (reported as GenericDevicePerf) • vmmemory - virtual machine memory performance data (reported as VirtualMachineMemoryPerf) • vmcpu - virtual machine CPU performance data that applies to all virtual CPUs (reported as VirtualMachineCPUPerf)

PerfType

PerfType describes the specification of a particular statistic that is gathered from a particular (single) source.

Name	Type	Description
key	xsd:string	Key of this sample.
type	PerfStatType	Type of performance statistic.
device	xsd:string	Name of the object for which you are gathering performance statistics.

Permission

Permission is a field of Item datatype and describes the access permissions for the Item.

Name	Type	Description
key	xsd:string	The key associated with this Permission object.
rights	xsd:string	One of the following access rights: Browse, Interact, Configure, or Administer.

PermissionList

PermissionList provides a list of permissions.

Name	Type	Description
perm	Permission[ ]	(Optional) List of permissions.

Property

A property is a simple name-value pair, where the value is a string.

Note: In general, the Property datatype can be modified by PutUpdates. However, this is dependent upon whether or not the datatype containing the Property allows the PutUpdates operation.

For example, the GenericDeviceInfo datatype (GenericDeviceInfo on page 60) is not modifiable by PutUpdates. Therefore, in this case, Property is not modifiable. However, because the

VirtualMachineInfo datatype can be modified by PutUpdates, customProperty is modifiable for this datatype.

Name	Type	Description	Modifiable by PutUpdates
key	xsd:string	Property key.	Yes
val	xsd:string	Property value.	Yes

RemovableDeviceInfo

RemovableDeviceInfo describes the parameters for devices that may be disconnected from the virtual machine.

Name	Type	Description	Modifiable by PutUpdates
startConnected	xsd:boolean	Indicates whether or not the removable device should be connected at startup.	Yes (can be set when the virtual machine is powered on, but is only applied when the virtual machine is restarted)
allowGuestControls	xsd:boolean	Indicates whether or not the guest operating system should be able to control the removable device.	Yes

ServiceStat

ServiceStat provides cumulative statistics for the Web service for all operations since it was started. The averages are exponentially-weighted moving averages for the respective time periods: one minute, five minutes or fifteen minutes.

Name	Type	Description
totalSuccessfulOps	xsd:long	Total number of successful operations.
totalFailedOps	xsd:long	Total number of failed operations.

Name	Type	Description
oneMinAverageOpsPerSec	xsd:float	Average number of operations per second, over one minute.
fiveMinAverageOpsPerSec	xsd:float	Average number of operations per second, over five minutes.
fifteenMinAverageOpsPerSec	xsd:float	Average number of operations per second, over fifteen minutes.

SystemInfo

Describes the version information for a host, virtual machine, or software component.

Name	Type	Description
name	xsd:string	Name of the system.
type	xsd:string	Type of system.
version	xsd:string	Version of the system as a string.
build	xsd:int	(Optional) Additional information on system.

Task

Task describes the task.

Name	Type	Description
operationName	xsd:string	Operation name of the task; for example, PutUpdates, or CreateTemplate, or any operation that may be scheduled.
cause	xsd:string	Agent causing the task.
schedule	xsd:string	(Optional) Schedule causing the task.
entity	xsd:string	Task operates on this object (VirtualCenter entity).
eventCollector	xsd:string	(Optional) Handle for the event collector of this task.

Name	Type	Description
currentState	TaskRunState	(Optional) State of the task. Takes one of the following: • scheduled • starting • running • completed • failed • killed
percentCompleted	xsd:float	(Optional) The percentage of the task that is complete. Possible values range from 0.0 (when "currentState" is "starting") to 100.0 (when "currentState" is "completed", "failed", or "killed". When the "currentState" is "running", the value increases from 0.0 to 100.0
faultReturn	FaultInfo	(Optional) If applicable, this is the error returned by the task operation. See FaultInfo on page 57..
allowCancel	xsd:boolean	Boolean indicating whether or not this task can be cancelled.
queueTime	xsd:dateTime	Specifies when VirtualCenter queued the task to run.

TaskRunState

TaskRunState describes the task status.

Name	Type	Description
val	xsd:string	Takes one of the following values: • scheduled • starting • running • completed • failed • killed

TaskSchedule

TaskSchedule describes a schedule for a task.

Name	Type	Description
name	xsd:string	Name of the scheduled task.
enabled	xsd:boolean	Boolean indicating whether or not the schedule is enabled.
entity	xsd:string	The main object that is operated upon; typically, one of the parameters.
recurrence	xsd:anyType	The recurrence of the task. Takes one of the following values: • OneTimeTask • AfterStartupTask • HourlyTask • DailyTask • WeeklyTask • MonthlyTask
eventCollector	xsd:string [ ]	(Optional) Handle for event collector of this schedule.
nextRun	xsd:dateTime	(Optional) The next time the task will start running. If this is not present, then there are no more tasks to run from this task schedule.
lastRun	xsd:dateTime	(Optional) From this schedule, the last time a task ran. If not present, then no tasks were run.
currentState	TaskRunState	(Optional) State of either the running task or the last task caused by this schedule.
faultReturn	FaultInfo	(Optional) The error returned by the last task caused by this schedule.

TaskScheduleSpec

TaskSchedule describes the specification for a task schedule. Use this datatype when creating a scheduled task using the Create operation (Create on page 148).

Name	Type	Description
name	xsd:string	Name of the task schedule.

Name	Type	Description
operationName	xsd:string	Name of the scheduled API operation; for example, StartVM or StopVM.
parameter	KeyedValue [ ]	(Optional) Parameters for the operation. This is a key-based array of key-value paris, where key is the name of the parameter and value is the value of the parameter.
recurrence	xsd:anyType	Recurrence of this task schedule.

Template

Template provides information about a template.

Name	Type	Description
name	xsd:string	Name of the template.
description	xsd:string	Description for the template.
guestOS	xsd:string	Guest operating system for the template.
memMb	xsd:int	Size of memory, in MB.
diskMb	xsd:int	Total disk size, in MB.
numNic	xsd:int	Number of virtual network interface cards for the template.
datastore	xsd:string	Datastore, on which the template is stored.

TemplateSpec

TemplateSpec provides the specifications for creating a template.

Name	Type	Description
name	xsd:string	Name of the template.
description	xsd:string	Description for the template.
datastore	xsd:string	(Optional) Indicates the datastore where the template files should be stored.

Update

An Update object describes the changes to a vHandle.

Name	Type	Description
handle	xsd:string	Handle that refers to the object.
vHandle	xsd:string	vHandle that refers to the object.
change	Change[ ]	Gives the actual set of changes.

UpdateList

UpdateList provides a list of updates.

Name	Type	Description
update	Update[ ]	List of updates.

VHandleList

VHandleList provides a list of vHandles.

Name	Type	Description
vHandle	xsd:string[ ]	List of vHandles.

VMMemoryLimits

VMMemoryLimits is a keyed array that provides the amount of memory available for creating virtual machines on the host for a given number of CPUs on the virtual machine. The first element of this array provides the memory limit for one CPU virtual machines, the second entry for two CPU virtual machines, and so on.

Name	Type	Description
key	xsd:string	Internal server key for the array element.
cpuCount	xsd:int	Number of virtual machine CPUs, for which this limit applies.

Name	Type	Description
maxMb	xsd:int	Maximum amount of memory for the virtual machine per cpuCount, in MB.

ViewContents

ViewContents identifies an object and the value of the object identified by the handle.

Name	Type	Description
handle	xsd:string	Handle that refers to the object.
vHandle	xsd:string	vHandle that refers to the object.
body	xsd:anyType	XML document that describes the object. We also refer to this XML document as the value of the object.

ViewInfo

ViewInfo is the datatype returned by a GetInfo operation on an object handle.

Name	Type	Description
parent	xsd:string	Handle that refers to the object's parent. If the object happens to be root (/), then the parent handle is the object handle.
name	xsd:string	Name of the object, relative to its parent.
type	xsd:string	Type of object.
perm	Permission	(Optional) Permissions for this object.

VirtualCPUInfo

VirtualCPUInfo provides information about the CPUs in the virtual machine.

Name	Type	Description	Modifiable by PutUpdates
count	xsd:int	Number of virtual CPUs.	Yes, only when the virtual machine is powered off. (ESX Server only)

Name	Type	Description	Modifiable by PutUpdates
controls	CPUControls	(Optional) Virtual CPU resource allocation information. For more information, see CPUControls (ESX Server Only) on page 39.	Yes (ESX Server only)

VirtualDisk

Provides information about virtual disks.

Name	Type	Description	Modifiable by PutUpdates
diskInfo	VirtualDiskInfo[ ]	(Optional) Disk information, one per virtual disk.	Yes
controls	DiskControls	(Optional) Resource management for a virtual machine's disk(s). For more information, seeDiskControls (ESX Server Only) on page 50.	Yes

VirtualDiskDestination

VirtualDiskDestination is an array that allows different disks to be placed on different datastores when moving a virtual machine. In this release, the VirtualDiskDestination array can have only one element, and all disks are placed on the specified datastore.

Name	Type	Description
key	xsd:string	Virtual disk key from the VirtualDiskInfo datatype for a virtual machine. This key specifies a disk that will be moved. **Note:** In this release, all disks are moved to the same dataLocator and clients may leave this key unspecified as it is not used.
dataLocator	xsd:string	Path describing where the virtual disk file should be stored.

VirtualDiskInfo

VirtualDiskInfo provides information about the parameters controlling the creation of a virtual disk on a virtual machine.

Name	Type	Description	Modifiable by PutUpdates
key	xsd:string	Key of the virtual disk.	No

Name	Type	Description	Modifiable by PutUpdates
name	xsd:string	Logical name of the virtual disk; for example, "HardDisk1". **Note:** In this release, the CreateVirtualDisk operation ignores this name.	Yes
dataLocator	xsd:string	(Optional) Path to the datastore on which the disk is created. If this is not specified, the Web service picks the datastore. If your virtual machine has raw disk mappings, then set this field to the datastore on which the raw disk mapping will be created.	Yes
controllerType	DiskControllerType	Type of controller; either "scsi" or "ide" for CD-ROM devices and virtual disks, and "floppy" for floppy drives. You can specify the controllerType by using the PutUpdates operation when creating the device. **Note:** After creating a virtual disk or CD-ROM device, you cannot change the controllerType with a PutUpdates call. If you want to change the controllerType, then delete the existing CD-ROM device or virtual disk, and create a new one with a different controllerType.	Yes
controllerId	xsd:int	Controller that should be used for this virtual disk.	Yes
deviceNumber	xsd:int	Specifies the device on the controller.	Yes
adapterType	DiskAdapterType	(Optional) This field is only used for SCSI disks and is either "busLogic" or "lsiLogic". In this current release, the adapterType stays at the default value of "busLogic". Setting the adapterType in the VirtualDiskInfo datatype to "lsiLogic" is ignored.	No
diskType	DiskType	Specifies the disk structure.	Yes
mode	VirtualDiskMode	(Optional) Indicates the virtual disk mode. The default value is "persistent".	Yes (only when the virtual machine is powered off)

Name	Type	Description	Modifiable by PutUpdates
writethru	xsd:boolean	(Optional) Indicates whether or not to persist all writes to the disk synchronously.The default is true.	Yes (only when the virtual machine is powered off)
legacy	xsd:boolean	(Optional) Boolean flag for legacy mode (copy-on-write or COW) disks.	No
removable	RemovableDeviceInfo	(Optional) Describes parameters for devices that may be disconnected from the virtual machine.	Yes (only when the virtual machine is powered off)
hostDev	xsd:string	(Optional) Host adapter and device where the virtual disk resides. This field is only relevant for floppy, CD-ROM, and SCSI disks with a diskKind of "device". If you are creating a virtual machine with raw disk mapping, then set this field to the volume name of the raw disk, for example, `vmhba0:0:0`.	Yes (only for floppy and CD-ROM disks. Can be set when the virtual machine is powered on, but is only applied when the virtual machine is restarted.)
exclusive	xsd:boolean	(Optional) Flag indicating exclusive access for the virtual disk. This is only relevant for CD-ROM devices and is currently ignored for other devices.	Yes (only for CD-ROM devices, when the virtual machine is powered off)
raw	xsd:boolean	(Optional) Flag indicating whether or not to persist all writes to the disk synchronously. This is only relevant for CD-ROM devices and is currently ignored for other devices.	Yes (only for CD-ROM devices, when the virtual machine is powered off)

VirtualDiskMode

VirtualDiskMode provides information about the disk mode of the virtual disk.

Name	Type	Description
val	xsd:string	Takes one of the following values: • append • independentNonpersistent • independentPersistent • nonpersistent • persistent • undoable

VirtualHardware

VirtualHardware provides information about the processor, memory, network interface, disk, and devices on a virtual machine.

Name	Type	Description	Modifiable by PutUpdates
cpu	VirtualCPUInfo	Processor information.	No
memory	VirtualMemoryInfo	Memory information.	No
net	VirtualNetworkInfo	(Optional) Virtual network interface information.	Yes
disk	VirtualDisk	(Optional) Virtual disk information.	Yes
floppy	VirtualDiskInfo[ ]	(Optional) Floppy disk information, one per virtual floppy drive.	Yes
cd	VirtualDiskInfo[ ]	(Optional) CD-ROM disk information, one per virtual CD-ROM.	Yes
device	GenericDeviceInfo[ ]	(Optional) Information on generic devices.	No

VirtualMachine

VirtualMachine provides system information, virtual hardware, status, performance data, and event information for a virtual machine.

Name	Type	Description
info	VirtualMachineInfo	System information for the virtual machine.
guestInfo	GuestInfo[ ]	(Optional) Variables exported by the guest operating system. For more information on these variables, see Virtual Machine Configuration Variables on page 201.
hardware	VirtualHardware	Virtual hardware for the virtual machine.
state	VirtualMachineState	Current state of the virtual machine.
path	xsd:string	Full path for a VirtualCenter object; for example, `/vcenter/Farms/Farm1/VMGroup1/VM1`. **Note:** Any unescaped `/` (slash) character used in this path parameter is regarded as a delimiter between path segments. Any `/` (slash) character not intended for use as a delimiter, will be escaped. Similarly, any `%` (percent) character used in this path parameter will be escaped, unless it is used to start an escape sequence.

VirtualMachineCPUPerf (ESX Server Only)

VirtualMachineCPUPerf details information on virtual machine CPUs, that apply to all virtual CPUs.

Name	Type	Description
effectiveMin	xsd:long	(Optional) Effective guaranteed minimum processor availability (in milliseconds) for a virtual machine, that is the greater value, based on CpuControls:min or CpuControls:shares.
extra	xsd:long	(Optional) Cumulative processor availability (in milliseconds) above "effectiveMin" for a virtual machine.

VirtualMachineGroup

A VirtualMachineGroup is similar to a Container, as it also contains an array of Items. VirtualMachineGroup represents VMware VirtualCenter Virtual Machine Groups that can contain Virtual Machine Groups, virtual machines, or both.

Name	Type	Description
item	Item[ ]	(Optional) Array of Items that can be of type VirtualMachineGroup and VirtualMachine.

VirtualMachineInfo

VirtualMachineInfo provides system information for a virtual machine.

Name	Type	Description	Modifiable by PutUpdates
uuid	xsd:string	SMBIOS (system management BIOS) UUID for the virtual machine.	No
name	xsd:string	User-defined name for the virtual machine.	No
guestOS	xsd:string	Guest operating system for this virtual machine.	Yes
file	xsd:string	Configuration file name for the virtual machine.	No
cfgVersion	xsd:string	Configuration version.	No
hwVersion	xsd:string	Virtual hardware version.	No

Name	Type	Description	Modifiable by PutUpdates
redoLogLocation	xsd:string	(Optional) File path of the redo logs for the virtual machine. The default redoLogLocation is different for ESX Server and GSX Server. If a non-default value is specified, then this field has a value other than an empty string. • ESX Server — Each virtual disk can be on a different volume; similarly, each redo log directory can also be on a different volume. If the redo log directory field is empty, then the default location of the redo log is on the same volume as the virtual disk file. • GSX Server — Users can specify the redo log directory in the virtual machine settings editor by choosing **VM > Settings**. On the **Options** tab, select **General**, then under **Working directory**, enter the directory name. The default location is the same directory as the virtual machine's configuration file (`.vmx`).	Yes (only when the virtual machine is powered off)
suspendDirectory	xsd:string	(Optional) Location of the suspend directory for the virtual machine.	Yes (only when the virtual machine is powered off)
disableAcceleration	xsd:boolean	Setting to turn off video acceleration for the GUI.	Yes (can be set when the virtual machine is powered on, but is only applied when the virtual machine is restarted)
enableLogging	xsd:boolean	Setting to enable logging for the virtual machine.	Yes (only when the virtual machine is powered off)
autostart	Autostart	(Optional) Setting to start the virtual machine automatically when host is started.	Yes (only when the virtual machine is powered off)

Name	Type	Description	Modifiable by PutUpdates
autostartCompleteAfter	xsd:int	(Optional) Number of seconds after the host starts, when the virtual machine's autostart is assumed to be complete.	Yes (only when the virtual machine is powered off)
autostartCompleteWhen ToolsStarted	xsd:boolean	(Optional) Setting that indicates whether or not a virtual machine's autostart should be considered to be complete, when VMware Tools have started.	Yes (only when the virtual machine is powered off)
autostop	Autostop	(Optional) Setting to stop or suspend the virtual machine when the host is shut down.	Yes (only when the virtual machine is powered off)
autostopCompleteAfter	xsd:int	(Optional) Number of seconds when the virtual machine's autostop is assumed to be complete.	Yes (only when the virtual machine is powered off)
tools	VirtualMachineTools	(Optional) Information about VMware Tools.	Yes
customPropertyDef	xsd:string	Handle to the list of all user-defined properties in VirtualCenter.	No
config	Property[]	(Optional) Configuration parameters for the virtual machine. These parameters are listed following this table, including whether or not they are modifiable. **Note:** Although the config Property keys are not modifiable, some of the config Property values are, as indicated in the following table.	Some config Property values, see the next table.
customProperty	Property[]	(Optional) Name/value pairs associated with the custom fields defined in VirtualCenter. Clients may update the existing values by using the PutUpdates operation and can add new properties only through the use of VirtualCenter.	Yes

The following table lists the config parameters for the VirtualMachineInfo data type, including whether or not they are modifiable. Any other config parameters are for internal use only and cannot be updated.

Name	Description	Modifiable by PutUpdates
annotation	User-defined string description for the virtual machine.	Yes
autostartOrder	Order in which this virtual machine should start on a particular host, when all virtual machines are started together.	Yes
autostopOrder	Order in which this virtual machine should stop on a particular host, when all virtual machines are stopped together.	Yes
location	Encoded location of the virtual machine's configuration file.	No
logFileName	Log file name for the virtual machine.	Yes
nvramFileName	File containing the BIOS and NVRAM settings for the virtual machine.	No
migrateHostlog	Log file that is updated when a virtual machine migrates.	No
checkpointCptConfigName	Snapshot file name.	Yes
exitAtPowerOff	GUI setting to exit after the virtual machine powers off.	Yes
exitOnCliHlt	GUI setting to exit on command line halt.	Yes
fullScreenAtPowerOn	GUI setting to switch to full screen mode on power on of the virtual machine.	Yes
powerOnAtStartup	Power on the virtual machine when it is opened.	Yes
snapshotAction	Snapshot setting: • default • prompt • keep • autoRevert • autoCommit	Yes
snapshotDisabled	Setting to disallow snapshots.	Yes
snapshotLocked	Setting to prevent previous snapshots from being overwritten.	Yes

VirtualMachineMemoryPerf

VirtualMachineMemoryPerf provides memory performance data for virtual machines. All sizes are reported in kilobytes (KB), since KB gives more detail than sizes in megabytes (MB).

Name	Type	Description	Applies to
size	xsd:int	(Optional) Current memory allocation, in KB.	ESX Server GSX Server
active	xsd:int	(Optional) Current working set estimate, in KB. GSX Server publishes a value for active, but it is the same number as size (preceding row in this table).	ESX Server only

VirtualMachineMemoryPerf2 (ESX Server Only)

VirtualMachineMemoryPerf2 augments the VirtualMachineMemoryPerf datatype. It includes 8 additional fields, shown in the following table.

Note: This new performance statistic field will only be available to the client by using the new QueryPerfData2 operation.

Name	Type	Description
size	xsd:int	(Optional) Current memory allocation, in KB.
active	xsd:int	(Optional) Current working set estimate, in KB.
shared	xsd:int	(Optional) Current amount of memory shared via transparent page sharing, in KB.
swapped	xsd:int	(Optional) Current amount of memory reclaimed by paging out to VMkernel swap files.
swapTarget	xsd:int	(Optional) Target amount of memory the host is trying to reclaim by paging out to VMkernel swap files. The host swaps more when swapTarget > swapped.
swapIn	xsd:int	(Optional) Cumulative amount of memory paged in from VMkernel swap files, in KB.
swapOut	xsd:int	(Optional) Cumulative amount of memory paged out to VMkernel swap files, in KB.
memctl	xsd:int	(Optional) Current amount of memory reclaimed by ballooning, in KB.

Name	Type	Description
memctlTarget	xsd:int	(Optional) Target amount of memory the host is trying to balloon for the virtual machine. The balloon driver allocates more memory or "inflates" when memctlTarget > memctl; it releases memory or "deflates" when memctlTarget < memctl.
overhead	xsd:int	(Optional) Current overhead memory size, in KB. Overhead memory is memory consumed for virtualization.

VirtualMachineSpec

VirtualMachineSpec provides specifications about the virtual machine.

Name	Type	Description
host	xsd:string	(Optional) The virtual machine belongs to this host.
guestOS	xsd:string	Key value that specifies the guest operating system in the virtual machine. The list of supported guest operating systems for a particular host machine is available in the HostInfo.configLimits.guestOS data structure.
file	xsd:string	(Optional) Location of the virtual machine configuration file.
hardware	VirtualHardware	(Optional) Specifies the virtual hardware for the virtual machine. When using the CloneVM operation, only the cpu, memory and net fields must be provided. Passing the disk, floppy, cd, or device fields can cause an exception.

VirtualMachineState

VirtualMachineState provides information about a virtual machine's status.

Name	Type	Description
host	xsd:string	(Optional) Handle from the virtual machine to the host where the virtual machine is active.
bootTime	xsd:dateTime	Time when the virtual machine was last booted. Not used in this current release.

Name	Type	Description
status	xsd:string	Current execution state of the virtual machine. This can take only one of the following values: • poweredOn • poweredOff • suspended • unknown The first three states in the preceding list are stable states. A newly created or discovered virtual machine remains in the unknown state until all its interior nodes are populated. Its state is then changed to poweredOff. During various power operations and migrations, the virtual machine may go through some transient states, which are listed in the detail field in the next row.

Name	Type	Description
detail	xsd:string	Contains additional information regarding the virtual machine state. Clients should monitor the status field only as the detail field may change in future implementations of the Web service. This field takes any of the following values: • _beingcreated — initial state • _coldmigrating — virtual machine is migrating while poweredOff • _connecting — transient state • _disconnecting — transient state • _reconnecting — transient state • _emigrating — virtual machine is being migrated to the destination host • _hotmigrating — virtual machine is migrating while poweredOn • _removing — transient state • _resetting — transient state • _resuming — transient state • _starting — transient state • _stopping — transient state • _suspending — transient state • disconnected — the virtual machine's host is not connected • initial • orphaned – the virtual machine's host is not known • poweredOff • poweredOn • suspended
migrating	xsd:boolean	Indicates whether or not the virtual machine is migrating to a different host.
connected	xsd:boolean	This boolean is set to true when the host is in the connected state in VirtualCenter.
eventCollector	xsd:string	Handle for event collector of this virtual machine.
msgWaiting	msgWaiting	(Optional) Describes the message that is blocking the virtual machine, including the message text, the message ID, the possible answers, and the default answer.

VirtualMachineTools

VirtualMachineTools provides information about VMware Tools and scripts that execute in the guest operating system when a virtual machine's power state changes.

Name	Type	Description	Modifiable by PutUpdates
toolsVersion	xsd:boolean	(Optional) Version of VMware Tools installed on the guest operating system.	No
afterPowerOn	xsd:boolean	(Optional) Flag determines whether or not scripts should run after the virtual machine is powered on. If this boolean is set to true, then custom startup scripts (if there are any) run on the guest operating system after the virtual machine powers on.	Yes (only when the virtual machine is powered off)
afterResume	xsd:boolean	(Optional) Flag determines whether or not scripts should run after the virtual machine is resumed. If this boolean is set to true, then custom startup scripts (if there are any) run on the guest operating system after the virtual machine resumes.	Yes (only when the virtual machine is powered off)
beforeSuspend	xsd:boolean	(Optional) Flag determines whether or not scripts should run before the virtual machine is suspended. If this boolean is set to true, then custom startup scripts (if there are any) run on the guest operating system before the virtual machine is suspended, regardless of whether the soft flag is specified during the StopVM operation.	Yes (only when the virtual machine is powered off)
beforePowerOff	xsd:boolean	(Optional) Flag determines whether or not scripts should run before the virtual machine is powered off. If this boolean is set to true, then custom startup scripts (if there are any) run on the guest operating system before the virtual machine powers off, regardless of whether the soft flag is specified during the StopVM operation.	Yes (only when the virtual machine is powered off)

VirtualMachineToolsStatus

VirtualMachineToolsStatus indicates the current status of VMware Tools on the guest operating system.

Name	Type	Description
val	xsd:string	Shows one of the following values: • none • old • ok

VirtualMemoryInfo

VirtualMemoryInfo provides information about a virtual machine's memory.

Name	Type	Description	Modifiable by PutUpdates
sizeMb	xsd:int	Size information.	Yes (only when the virtual machine is powered off)
controls	MemoryControls	(Optional) Resource management information. For more information, see MemoryControls (ESX Server Only) on page 77.	Yes

VirtualNetworkAdapter

VirtualNetworkAdapter provides information about the virtual network adapter.

Name	Type	Description	Modifiable by PutUpdates
key	xsd:string	Key of the virtual network adapter.	No
name	xsd:string	Name of the virtual network adapter.	Yes
emulation	VirtualNetworkEmulation	Describes the network driver used to connect the virtual machine to the network.	Yes (only when the virtual machine is powered off)
mode	VirtualNetworkMode	Describes the virtual network.	Yes

Name	Type	Description	Modifiable by PutUpdates
device	xsd:string	The Device type of the adapter; for example, vmnic.	Yes
network	xsd:string	Network name for the Adapter; for example, Internal Network	Yes (can be set when the virtual machine is powered on, but is only applied when the virtual machine is restarted)
addressType	VirtualNetworkAddressType	Describes the type of network address.	Yes (only when the virtual machine is powered off)
address	xsd:string	MAC address of the adapter (physical adapter only). This field is empty for other types.	Yes (only when the virtual machine is powered off)
features	xsd:int	Additional features supported by the adapter	Yes

VirtualNetworkAddressType

VirtualNetworkAddressType describes the type of network address.

Name	Type	Description
val	xsd:string	Takes one of the following values: • none • manual • generated • assigned (for use with VMotion)

VirtualNetworkEmulation

VirtualNetworkEmulation describes the network driver used to connect the virtual machine to the network.

Name	Type	Description
val	xsd:string	Takes one of the following values: • vlance • vmxnet

VirtualNetworkInfo

VirtualNetworkInfo provides information about all the virtual network interfaces.

Name	Type	Description	Modifiable by PutUpdates
adapter	VirtualNetworkAdapter[ ]	(Optional) Information about each network interface.	Yes
controls	NetworkControls	(Optional) Share information. For more information, see NetworkControls (ESX Server Only) on page 81.	Yes

VirtualNetworkMode

VirtualNetworkMode describes the type of virtual network.

Name	Type	Description
val	xsd:string	Takes one of the following values: • bridged • custom • hostOnly • monitor • nat • vmkernel

WeekDay

WeekDay describes the days of the week.

Name	Type	Description
val	xsd:string	Takes one of the following values: • sunday • monday • tuesday • wednesday • thursday • friday • saturday

WeeklyTask

WeeklyTask describes a task that runs once every week or once every few weeks, depending on the interval you specify.

Name	Type	Description
hours	xsd:int	The task is scheduled at this hour of the day. For example, "20" specifies that this task should run at 8 p.m.
minutes	xsd:int	Specifies the minutes of the hour when the task is scheduled. For example, "12" specifies that this task should run 12 minutes after the hour.
dayOfWeek	WeekDay	The task is scheduled for this day of the week. Takes one of the following values: • sunday • monday • tuesday • wednesday • thursday • friday • saturday
interval	xsd:int	Specifies the number of weeks (interval) between scheduled tasks. For example, "2" specifies that this task should run every two weeks.

msgWaiting

msgWaiting describes a message that is blocking the virtual machine. Clients can unblock this virtual machine by using the AnswerVM operation.

Name	Type	Description
msg	xsd:string	(Optional) Question that the blocked virtual machine is asking.
id	xsd:string	(Optional) Internal server ID for the message.
choice	KeyedValue	(Optional) Array of possible answers for the message, where each answer is a key-value pair.
defaultChoiceIndex	xsd:int	(Optional) Integer that specifies which key-value pairs is the default answer to the message.

CHAPTER 4

Basic Operations

This chapter describes the following basic operations:

- Login on page 115
- Logout on page 117
- ResolvePath on page 118
- GetContents on page 120
- GetInfo on page 123

Each service request contains an operation name and may contain input parameters. The response, if successful, contains any specified output parameters, or error information, if a fault occurs during request processing.

If the service successfully processes a request, then it returns the HTTP status code 200 (OK). If the request fails, then the service returns an error code that is classified in as similar manner to HTTP error codes, and the body contains an XML document with the fault information.

Two special faults are possible for any request:

- UnknownRequest — the request specified an invalid operation name
- SystemFailure — the Web service encountered a system error during request processing. An example of a SystemFailure is if the service runs out of local machine resources (for example, memory) during processing.

Login

The Login operation specifies a new user and associated password to use for this client. The interpretation of user names is system-dependent. Typically, a Web service implementation authenticates the user name and password with the VirtualCenter service running on the same Windows machine as the Web service.

Permissions

Any client can call the Login operation.

Input Message – LoginRequestMsg

All communication between the client and the Web service uses secure HTTP (HTTPS) when a `/https` URL is used to an SSL port in the Web service, ensuring that the authentication information is encrypted and not sent in the clear.

Login		
Name	**Type**	**Description**
userName	xsd:string	User identifier for authentication on system running the Web service.
password	xsd:string	Plain text string that is the password for the user

A successful login request starts a new session for the username, that is uniquely identified by the session token issued by the Web service to the requesting client. The client is expected to pass this session token back, with each subsequent request. The Web service maintains the session token/user mapping, and assumes the identity of the user to check permissions on each request.

Example

```
<userName>joe</userName>
<password>rover</password>
```

Output Message – LoginResponseMsg

If the Login operation is successful, then the request returns successfully.

Fault

FaultInfo		
Name	**Type**	**Description**
kind	FaultKind	• NotApplicable — Service is not available. • PermissionDenied — userName or password supplied in the Login request is not recognized by the Web service.
code	xsd:int	Integer, which gives the error code that is classified in a manner similar to HTTP error codes.
info	xsd:string	See the FaultInfo datatype for details.
val	xsd:anyType	See the FaultInfo datatype for details.

Logout

The Logout operation ends the session identified by the session token. (The session token is automatically passed with the HTTP header of the Logout request.) This operation takes no parameters, has no results, and does not generate any faults.

Note that closing a connection is not necessarily equivalent to a logout, as a session token may work with new connections.

Permissions

Upon Logout, the client loses any access rights that it may have. Any client can perform a Logout request.

Input Message – LogoutRequestMsg

The Logout operation does not take any input parameters.

Output Message – LogoutResponseMsg

If the Logout operation is successful, then the request returns successfully.

Fault

FaultInfo		
Name	**Type**	**Description**
kind	FaultKind	• BadRequest — The request is not well-formed.
code	xsd:int	Integer, which gives the error code that is classified in a manner similar to HTTP error codes.
info	xsd:string	See the FaultInfo datatype for details.
val	xsd:anyType	See the FaultInfo datatype for details.

ResolvePath

The ResolvePath operation takes a path as its only input parameter and returns a handle to the object referenced by the path. An object handle is somewhat analogous to a file handle (descriptor) returned by a operating system similar to UNIX, when a file is opened using the "open" system call. Like a file handle, an object handle is a temporary handle that always refers to the same object.

Permissions

The client must have Browse rights on the object.

Input Message — ResolvePathRequestMsg

ResolvePath		
Name	**Type**	**Description**
path	xsd:string	Path to the object. Any unescaped / (slash) character used in this path parameter is treated as a separator between path segments. Clients must escape this / (slash) character if it is not intended to be a path segment separator. **Note:** Any % (percent) character used in this path parameter must be escaped, unless it is used to start an escape sequence. Clients may also escape any other characters in this path parameter.

Output Message — ResolvePathResponseMsg

ResolvePathResponse		
Name	**Type**	**Description**
returnval	xsd:string	Handle to the object referred to by the input path parameter.

Fault

FaultInfo		
Name	**Type**	**Description**
kind	FaultKind	• NotFound — Path does not point to a valid object. • PermissionDenied —Client does not have Browse permission for the object.

FaultInfo		
Name	**Type**	**Description**
code	xsd:int	Integer, which gives the error code that is classified in a manner similar to HTTP error codes.
info	xsd:string	See the FaultInfo datatype for details.
val	xsd:anyType	See the FaultInfo datatype for details.

GetContents

The GetContents operation retrieves the data for a particular object, such as a virtual machine, host, or Container. This operation takes an object handle as its only input parameter. The returned value is of type ViewContents, that represents the contents of the object at a certain point in time. The vHandle element of ViewContents is the versioned handle of that object (at a specific point in time) and body is the object data.

Permissions

The client must have Browse rights to the desired object.

Input Message – GetContentsRequestMsg

The client specifies the handle of the object that it wishes to retrieve. Containers, Farms, VirtualMachineGroups, Tasks, Events, Schedules, hosts, virtual machines, and performance data collectors (perf collectors) are all valid objects managed by the Web service. Each object is described by an XML document, which is the value of the object.

If you use the file system analogy:

- Container, Farm and VirtualMachineGroup are comparable to directories in a file system
- All other objects are comparable to files in a file system

GetContents		
Name	**Type**	**Description**
handle	xsd:string	Indicates the handle of interest to the client. The handle may refer to any of the valid object types managed by the Web service.

Output Message – GetContentsResponseMsg

The response to the GetContents request is the contents of object specified by the input parameter handle.

GetContentsResponse		
Name	**Type**	**Description**
returnval	ViewContents	• handle — Handle for the object. • vHandle — Versioned handle for the object. • body — XML document describing the object.

The following table lists the values for returnval for a GetContents operation on the specified handles. A ResolvePath operation on the given path is required to get the corresponding handle.

Path	GetContents returnval	Comments
`/vm`	Container	List of Items that are of type VirtualMachine.
`/host`	Container	List of Items that are of type Host.
`/vcenter`	Container	List of Items that are of type Container and Farm.
`/task`	Container	List of Items that are of type Task.
`/task/id`	Task	
`/schedule`	Container	List of Items that are of type TaskSchedule.
`/schedule/id`	TaskSchedule	
`/event`	Container	List of Items that are of type Container.
`/event/decls`	EventDeclList	List of Items that are of type EventDecl.
`/event/all`	Container	List of "filter" and "events" Items.
`/event/all/filter`	EventFilter	Filter that collects all events.
`/event/all/events`	EventCollection	Collection of all events.
`/perf`	Container	List of Items (default performance collectors plus user-created performance collectors) that are of type Container.
`/perf/<perfname>`	Container	List of "filter" and "stats" Items.
`/perf/<perfname>/filter`	PerfFilter	Specification of filter and other information for a performance collector.
`/perf/<perfname>/stats`	PerfCollection	Most recent statistics that the performance collector is collecting.
Handle identifying a Container in `/vcenter`	Container	List of Items that are of type Container and Farm.
Handle identifying a Farm in `/vcenter`	Farm	List of Items that are of type VirtualMachineGroup, Host, and VirtualMachine.
Handle identifying a VirtualMachineGroup in `/vcenter`	VirtualMachineGroup	List of Items that are of type VirtualMachineGroup and VirtualMachine.
Handle identifying a VirtualMachine in `/vcenter` or `/vm/UUID`	VirtualMachine	
Handle identifying a Host in `/vcenter` or `/host/hostname`	Host	
`/webservice`	Container	Contains information about the Web service release in SystemInfo.
`/template`	Container	List of Items that are of type Template.

Path	GetContents returnval	Comments
`/datastore`	DatastoreInfoList	List of Items that are of type DatastoreInfo.

Fault

FaultInfo		
Name	**Type**	**Description**
kind	FaultKind	• BadRequest — The request is not well formed. • HandleFault – The handle, passed as a parameter, has encountered errors. • OperationFailed — The server could not successfully complete this operation.
code	xsd:int	Integer, which gives the error code that is classified in a manner similar to HTTP error codes.
info	xsd:string	See the FaultInfo datatype for details.
val	xsd:anyType	See the FaultInfo datatype for details.

GetInfo

The GetInfo operation takes the handle of an object as its input parameter and returns the parent handle, name, type, and permissions for the object. If the object happens to be the root container (the handle for /), then the parent handle is also the object handle.

Permissions

The client must have Browse rights on the object.

Input Message — GetInfoRequestMsg

GetInfo		
Name	**Type**	**Description**
handle	xsd:string	Handle for the object of interest.

Output Message — GetInfoResponseMsg

The parent handle, name, type, and permissions for the object are returned.

GetInfoResponse		
Name	**Type**	**Description**
returnval	ViewInfo	The response specifies the ViewInfo for the object. • parent — Handle for the parent object. • name — Name of the object. • type — Type of the object. • perm — Permissions for the object.

Fault

FaultInfo		
Name	**Type**	**Description**
kind	FaultKind	• HandleFault – The handle, passed as a parameter, has encountered errors.
code	xsd:int	Integer, which gives the error code that is classified in a manner similar to HTTP error codes.
info	xsd:string	See the FaultInfo datatype for details.

FaultInfo		
Name	**Type**	**Description**
val	xsd:anyType	See the FaultInfo datatype for details.

CHAPTER 5

Element Management Operations

This chapter describes the following element management operations:

- StartVM on page 131
- AnswerVM on page 133
- StopVM on page 135
- ResetVM on page 137
- StopHost on page 139
- GetUpdates on page 141
- CancelGetUpdates on page 144
- PutUpdates on page 146
- Create on page 148
- CreateVirtualDisk on page 156
- Rename on page 158
- Delete on page 162

- QueryPerfData on page 164
- QueryPerfData2 on page 166
- ChangePermissions on page 168
- SnapshotVM on page 170
- ConsolidateVM on page 172
- RevertVM on page 174

Performing Element Management Operations

The Element Management operations enable the client program to perform the following functions.

Virtual Machine Namespace Management, Inventory and Discovery

Each virtual machine is assigned a universally unique identifier (UUID) upon creation. This UUID does not change during the lifetime of the virtual machine and is globally unique. When a virtual machine is cloned, a new UUID is generated. Clients can discover active, running virtual machines and inactive virtual machines (powered off or suspended). Clients can collect the following information for each virtual machine:

- Virtual machine guest operating system
- Virtual machine state (poweredOn, poweredOff, or suspended)
- Virtual machine IP address and host name
- Virtual machine configuration

Virtual Machine Power Operations

Clients may invoke the following power operations on the virtual machines:

- Power on
- Power off
- Suspend
- Resume

Virtual Machine Creation, Deletion and Configuration

Clients may create new virtual machines and delete existing ones. You may use a client to configure each virtual machine, including the following attributes:

- Name of the virtual machine
- CPU Configuration (CPU minimum, maximum, and shares, and CPU affinities)
- Memory Configuration (memory minimum, maximum, and shares)
- Disk Configuration (shares)
- Network Configuration (traffic shaping)
- Hardware Configuration (connect or disconnect devices, bind to NICs, connect to a disk, set disk mode, set display options, and so on)

Real-Time Performance Data Gathering

Clients may gather performance data for each virtual machine and for the host machine. This includes:

- CPU Utilization
- Memory Utilization
- Disk Utilization (read/write rates per disk controller)
- Network Utilization (read/write rates per NIC)

Event Notification

Clients may monitor for events of interest that include:

- Addition or removal of hardware components
- Virtual machine creation or deletion
- Virtual machine state changes (poweredOn, poweredOff, suspended, and so on)

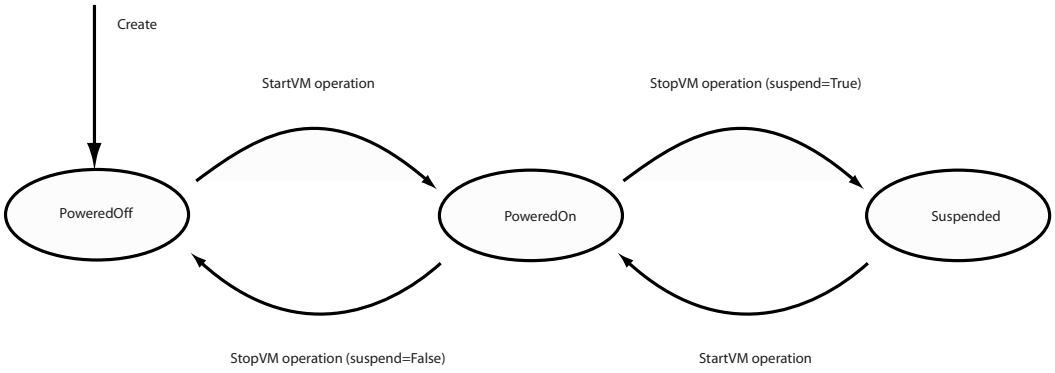
Virtual Machine State Transitions

A virtual machine may be in one of the following stable states:

- poweredOn
- poweredOff
- suspended

In addition to these stable states, a virtual machine may be in certain transient states; for example, when a powered on virtual machine is being powered off, it is temporarily in the _stopping state (see the detail field in VirtualMachineState on page 104), before settling down in the poweredOff state.

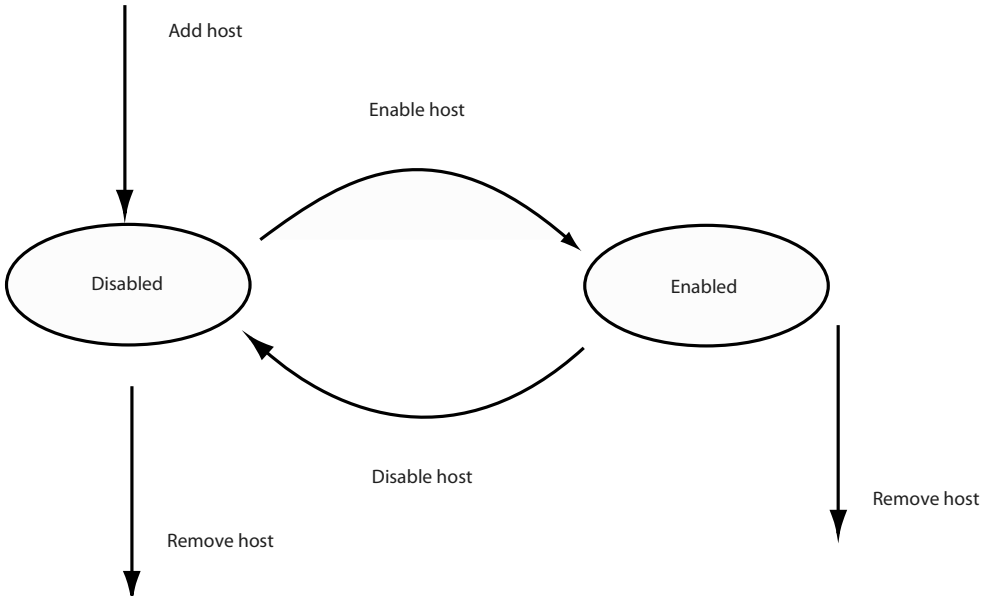
Note: Clients may not receive an update notification for these transient states. The Web service only returns the latest updates to the client from a GetUpdates call and all intermediate updates are lost.



Host State Transitions

A host machine may be in one of the following states:

- disabled
- enabled



StartVM

The StartVM operation initiates the process of starting a virtual machine or resuming a suspended a virtual machine. This operation returns a handle to an active task that performs the virtual machine power operation. Clients that wish to follow a virtual machines progress should monitor the currentState member of the Task datatype (to check when the task completes) and should use the GetUpdates operation on the virtual machine to look for changes to the state element.

Permissions

The client must have Interact rights on the target virtual machine.

Input Message — StartVMRequestMsg

StartVM		
Name	**Type**	**Description**
vm	xsd:string	Handle specifying the target virtual machine.

Output Message — StartVMResponseMsg

The request returns once the operation has been initiated. A task handle is returned back to the client. The client may monitor the task for progress of the operation. Alternatively, the client may monitor the state of the virtual machine for progress.

StartVMResponse		
Name	**Type**	**Description**
returnval	ViewContents	The response specifies the ViewContents of the task performing the StartVM operation. • ViewContents.handle – Handle for the task object. • ViewContents.vHandle – Versioned handle for the task object. • ViewContents.body – XML document of type Task that describes the task object.

Fault

FaultInfo		
Name	**Type**	**Description**
kind	FaultKind	• BadRequest — The request is not well-formed. • BadState — The virtual machine is inactive or already powered on. • HandleFault – The handle, passed as a parameter, has encountered errors. • NotApplicable — The specified handle does not refer to a virtual machine. • OperationFailed — The server could not successfully complete this operation.
code	xsd:int	Integer, which gives the error code that is classified in a manner similar to HTTP error codes.
info	xsd:string	See the FaultInfo datatype for details.
val	xsd:anyType	See the FaultInfo datatype for details.

AnswerVM

The AnswerVM operation allows the client to answer a message (question) posed by a virtual machine that is blocked and waiting for a response. The message is reported as part of the virtual machine's state during a GetContents or GetUpdates request. The message includes the following:

- msg — Text of the message (question).
- id — ID of the message.
- choice — Possible answers to the message, defined as an array of key-value pairs. Clients choose one of these key-value pairs to answer the message.
- defaultChoiceIndex (if present) — Index in the array of choices indicating the default choice that is `0` (zero). If the defaultChoiceIndex is not present, then there is no default answer available.

Permissions

The client must have Interact rights on the virtual machine.

Input Message — AnswerVMRequestMsg

AnswerVM		
Name	**Type**	**Description**
vm	xsd:string	Virtual machine that is asking the question.
choice	KeyedValue	Key-value pair for the selected choice. This choice is one of the entries included in the message.
id	xsd:string	ID of the message.

Output Message — AnswerVMResponseMsg

Upon success, an empty response message is returned.

Fault

Parameter	Type	Description
kind	FaultKind	• BadRequest — The request is not well-formed. • BadState — The virtual machine is not blocked on a message. • HandleFault – The handle, passed as a parameter, has encountered errors. • NotApplicable — The specified handle does not refer to a virtual machine. • OperationFailed — The server could not successfully complete this operation.
code	xsd:int	Integer, which gives the error code that is classified in a manner similar to HTTP error codes.
info	xsd:string	See the FaultInfo datatype for details.
val	xsd:anyType	See the FaultInfo datatype for details.

StopVM

The StopVM operation initiates the process of stopping or suspending a virtual machine. This operation returns a handle to an active task that performs the virtual machine power operation. Clients that wish to follow a virtual machines progress should monitor the currentState member of the Task datatype (to check when the task completes) and should use the GetUpdates operation on the virtual machine to look for changes to the state element.

Note: Suspending a virtual machine, immediately after it has been resumed, may sometimes fail. A small delay is required before the suspend operation commences.

Permissions

The client must have Interact rights on the target virtual machine.

Input Message — StopVMRequestMsg

StopVM		
Name	**Type**	**Description**
vm	xsd:string	Handle specifying the target virtual machine.
suspend	xsd:boolean	Flag, to indicate whether to suspend or stop the virtual machine.
soft	xsd:boolean	Flag, to indicate whether or not to have the guest operating system perform any additional action. When the virtual machine is being suspended, this flag indicates whether or not the guest operating system should go into standby mode. When the virtual machine is being stopped, this flag indicates whether or not the guest operating system should perform a graceful shutdown. **Note:** The VMware VirtualCenter Web Service waits five minutes for the guest operating system to shut down. If the guest operating system does not shut down within five minutes, then the Web service times out and report a timeout error.

Output Message — StopVMResponseMsg

The request returns once the operation has been initiated. A task handle is returned back to the client. The client may monitor the task for progress of the operation. Alternatively, the client may monitor the state of the virtual machine for progress.

StopVMResponse		
Name	**Type**	**Description**
returnval	ViewContents	The response specifies the ViewContents of the task performing the StopVM operation. • ViewContents.handle – Handle for the task object. • ViewContents.vHandle – Versioned handle for the task object. • ViewContents.body – XML document of type Task that describes the task object.

Fault

Parameter	Type	Description
kind	FaultKind	• BadRequest — The request is not well-formed. • BadState — The virtual machine is inactive or already powered off. • HandleFault – The handle, passed as a parameter, has encountered errors. • NotApplicable — The specified handle does not refer to a virtual machine. • OperationFailed — The server could not successfully complete this operation.
code	xsd:int	Integer, which gives the error code that is classified in a manner similar to HTTP error codes.
info	xsd:string	See the FaultInfo datatype for details.
val	xsd:anyType	See the FaultInfo datatype for details.

ResetVM

The ResetVM operation initiates the process of resetting a virtual machine, which also resets the virtual hardware. This operation returns a handle to an active task that performs the virtual machine power operation. Clients that wish to follow a virtual machine's progress should monitor the currentState member of the Task datatype (to check when the task completes) and should use the GetUpdates operation on the virtual machine to look for changes to the state element.

Permissions

The client must have Interact rights on the target virtual machine.

Input Message — ResetVMRequestMsg

ResetVM		
Name	**Type**	**Description**
vm	xsd:string	Handle specifying the target virtual machine, that must be in the poweredOn state, for this operation to take effect.

Output Message — ResetVMResponseMsg

The request returns once the operation has been initiated. A task handle is returned back to the client. The client may monitor the task for progress of the operation. Alternatively, the client may monitor the state of the virtual machine for progress.

ResetVMResponse		
Name	**Type**	**Description**
returnval	ViewContents	The response specifies the ViewContents of the task performing the ResetVM operation. • ViewContents.handle – Handle for the task object. • ViewContents.vHandle – Versioned handle for the task object. • ViewContents.body – XML document of type Task that describes the task object.

Fault

FaultInfo		
Name	**Type**	**Description**
kind	FaultKind	• BadRequest — The request is not well-formed. • BadState — The virtual machine is inactive, already powered off, or suspended. • HandleFault – The handle, passed as a parameter, has encountered errors. • NotApplicable — The specified handle does not refer to a virtual machine. • OperationFailed — The server could not successfully complete this operation.
code	xsd:int	Integer, which gives the error code that is classified in a manner similar to HTTP error codes.
info	xsd:string	See the FaultInfo datatype for details.
val	xsd:anyType	See the FaultInfo datatype for details.

StopHost

The StopHost operation permits the client to shut down or restart the host.

Permissions

The client must have Configure rights on the parent Farm of the host.

Input Message — StopHostRequestMsg

StopHost		
Name	**Type**	**Description**
host	xsd:string	Handle specifying the target host.
soft	xsd:boolean	If the soft flag is true, then the operation fails if there are running virtual machines. If the soft flag is false, then the host is immediately powered off or restarted, even if there are running virtual machines. This is equivalent to pressing the power button on the physical machine.
restart	xsd:boolean	If the restart flag is true, then the host is restarted. Otherwise, it is powered off.
reason	xsd:string	Reason for the StopHost request. This reason string is placed in the ESX Server logs as the reason for shutting down the host.

Output Message — StopHostResponseMsg

Upon success, an empty response message is returned.

Fault

FaultInfo		
Name	**Type**	**Description**
kind	FaultKind	• BadRequest — The request is not well-formed. • BadState — The host machine is inactive or powered off. • HandleFault – The handle, passed as a parameter, has encountered errors. • NotApplicable — The specified handle does not refer to a host machine. • OperationFailed — The server could not successfully complete this operation.

FaultInfo		
Name	**Type**	**Description**
code	xsd:int	Integer, that gives the error code that is similarly classified to HTTP error codes.
info	xsd:string	See the FaultInfo datatype for details.
val	xsd:anyType	See the FaultInfo datatype for details.

GetUpdates

A client executes the GetUpdates operation, passing in an array of vHandles as a parameter and retrieving the change(s) in the object identified by any vHandle. Only the change(s), or the "delta" is returned as a `diff` of the current XML document. The change(s) reflect the difference between the object as is currently maintained by the Web service, compared with the original XML document initially received by the client as part of the GetContents() operation.

The GetUpdates request is a data collection or a monitoring request depending upon the value of the input wait parameter. If wait is TRUE, then the Web service does not send a response until at least one object, that is included in the input vHandleList parameter, changes. However, the HTTP 1.1 protocol requires that the service respond to an outstanding request before processing a new request on the same client connection. When the Web service receives a new client request on a client HTTP 1.1 connection, on which there already is a blocking GetUpdates pending, the blocking GetUpdates returns with an "Interrupted" FaultKind. Clients can use the CancelGetUpdates operation to cancel a pending GetUpdates operation in the same session.

Note: A client may have multiple GetUpdates or PutUpdates (through independent threads and connections) pending at any given time.

Permissions

The client must have Interact rights on the specified handles.

Input Message — GetUpdatesRequestMsg

GetUpdates		
Name	**Type**	**Description**
vHandleList	VHandleList	The client is interested in monitoring this list of vHandles for changes and updates.
wait	xsd:boolean	Indicates whether or not the client is willing to wait for updates.

Output Message — GetUpdatesResponseMsg

GetUpdatesResponse		
Name	**Type**	**Description**
returnVal	UpdateList	Each entry in the UpdateList comprises a list of Update elements. An Update element contains a handle, the versioned handle, and the list of changes for the object referenced by the handle.

There are four different kinds of change elements:

- edit — Value is text to replace in the specified element
- ins — Value is one or more XML elements to insert into the document
- repl — Value is one or more XML elements to replace in the document
- del — Deletes an element and its children from the document
- move — Object is being renamed or moved in the hierarchy

See Change on page 41 and ChangeOp on page 43 for detailed information about these change elements.

The returned UpdateLIst may be different from the input vHandleList. However, it can only be as large as the list that you passed in (although it could be smaller). Within each update, there can be more than one change.

Fault

FaultInfo		
Name	**Type**	**Description**
kind	FaultKind	• BadRequest — The request is not well formed. • HandleFault – The handle, passed as a parameter, has encountered errors. • Interrupted – One or more of the vHandles in the request is now invalid. • OperationFailed — The server could not successfully complete this operation.
code	xsd:int	Integer, which gives the error code that is classified in a manner similar to HTTP error codes.
info	xsd:string	See the FaultInfo datatype for details.
val	xsd:anyType	See the FaultInfo datatype for details.

CancelGetUpdates

The CancelGetUpdates operation cancels all pending and future GetUpdates operations for the specified vHandle in a single session. Any GetUpdates calls with the specified vHandle throw a fault, with FaultInfo containing a kind of "HandleFault". The 'val' field of the FaultInfo structure contains a HandleFaultList. One of the entries in this list corresponds to the cancelled vHandle and includes an element named nextVHandle.

A client should use this nextVHandle in subsequent calls to GetUpdates, for the object whose vHandle was specified in the CancelGetUpdates operation. The cancelled vHandle continues to return a HandleFault if it is passed in a GetUpdates operation.

Permissions

The client must have Browse rights on the specified handles.

Input Message — CancelGetUpdatesRequestMsg

CancelGetUpdates		
Name	**Type**	**Description**
vHandle	xsd:string	The client is cancelling the GetUpdates operation on this versioned handle. Clients must use nextVHandle for any subsequent GetUpdates operations.

Output Message — CancelGetUpdatesResponseMsg

Upon success, an empty response message is returned.

Fault

FaultInfo		
Name	**Type**	**Description**
kind	FaultKind	• BadRequest — The request is not well formed. • HandleFault – The handle, passed as a parameter, has encountered errors. • Interrupted – The vHandle in the request is now invalid. • OperationFailed — The server could not successfully complete this operation.
code	xsd:int	Integer, which gives the error code that is classified in a manner similar to HTTP error codes.

FaultInfo		
Name	**Type**	**Description**
info	xsd:string	See the FaultInfo datatype for details.
val	xsd:anyType	See the FaultInfo datatype for details.

PutUpdates

The PutUpdates operation requests one or more changes to data values. A client executes the PutUpdates operation, passing in a vHandle or a handle and an XML document `diff` (showing only changes) to the object referred to by the vHandle. Clients should not pass in the complete XML document describing the updated object. By passing in only the `diff` (the changes), the client minimizes the amount of data that it must send to the Web service.

A client should use a handle instead of a vHandle when it wishes to update an object value, regardless of any other updates that may have occurred concurrently. If a client uses a vHandle, the Web service server first checks to see if any other updates have occurred since the vHandle was obtained. The Web service server rejects the PutUpdates operation if there is any possibility of overwriting newer changes with older changes.

Permissions

The client must have Configure rights to all the specified handles.

Input Message — PutUpdatesRequestMsg

Clients can pass to the PutUpdates operation only one ChangeReq, which can either contain a handle or a vHandle, for unversioned or versioned updates. Each ChangeReq can contain multiple changes. If the Web service returns a fault to a PutUpdates operation, the client should assume that none of the changes succeeded. When using the SDK with VirtualCenter 1.1 and 1.2, we recommend that clients update virtual machine resource settings (including memory and CPU shares) in a separate PutUpdates call from other virtual machine updates.

- If clients call PutUpdates with a vHandle, then the Web service always returns all changes made to an object, including the changes made by the PutUpdates call itself.
- If clients call PutUpdates with a handle, then success is indicated by an empty response, with no changes returned to the client.

A client may have multiple pending GetUpdates or PutUpdates operations (through independent threads and connections) at any given time.

Note: Clients cannot specify two or more devices, of the same type (for example, two NICs or two floppy drives) in a single PutUpdates call on a virtual machine.

Instead, specify one of the devices (for example, one NIC) in the first PutUpdates call, then specify the additional device (second NIC) by using a second PutUpdates call on the same virtual machine.

PutUpdates		
Name	**Type**	**Description**
req	ChangeReqList	List of change requisitions (ChangeReq). A ChangeReq comprises a handle (or a vHandle) and an array of Change types as described in Change on page 41.

Output Message — PutUpdatesResponseMsg

PutUpdatesResponse		
Name	**Type**	**Description**
returnval	UpdateList	Each entry in the UpdateList comprises the handle, vHandle and the changes. The changes that are returned to a client are a superset of the requested changes, because the client also gets back other changes, that are the result of actions made by other clients.

Fault

FaultInfo		
Name	**Type**	**Description**
kind	FaultKind	• BadRequest — The request is not well-formed. • ChangeConflict – During a PutUpdates operation, the Web service is unable to accept the changes that the client sent over, because the vHandle on the client does not match the Web service, and the changes conflict with another change already made on the Web service. The client should run the GetContents operation on the object again, then retry the change. • HandleFault – The handle, passed as a parameter, has encountered errors. • OperationFailed — The server could not successfully complete this operation.
code	xsd:int	Integer, which gives the error code that is classified in a manner similar to HTTP error codes.
info	xsd:string	See the FaultInfo datatype for details.
val	xsd:anyType	See the FaultInfo datatype for details.

Create

The Create operation requests the creation of a new object of a given type. Clients may create the following objects with this Create operation:

- VirtualMachine
- Host
- Container, Farm, or VirtualMachineGroup
- TaskSchedule
- PerfCollector
- Event collector

Note: Clients cannot use the Create operation to create a new virtual disk. Use the CreateVirtualDisk operation instead. See CreateVirtualDisk on page 156.

Permissions

The client must have Configure rights for the Container of the new object.

Input Message — CreateRequestMsg

Create		
Name	**Type**	**Description**
handle	xsd:string	Handle of the Container for the new object. This may be one of the following: • Handle for `/vm`, a `/vcenter` Farm, or a `/vcenter` VirtualMachineGroup for creating VirtualMachine. If the handle for `/vm` is specified, then a corresponding entry is also made in the Farm of the host on which the virtual machine is being created. • Handle for `/host` or a `/vcenter` Farm for creating Host. If the handle for `/host` is specified, a corresponding entry is made in the default Farm in the `/vcenter` directory, `/vcenter/Default Farm`. If the `/vcenter/Default Farm` directory is removed from VirtualCenter, then creating a host in `/host` may fail. **Note:** Creating and deleting hosts in quick succession may sometimes fail. • Handle for a `/vcenter` Container for creating Container or Farm. • Handle for a `/vcenter` Farm or `/vcenter` VirtualMachineGroup for creating VirtualMachineGroup. • Handle for `/schedule` to create TaskSchedule. • Handle for `/perf` to create PerfCollector.
name	xsd:string	Name for the new object. When the type is VirtualMachine and the handle is `/vm`, then this parameter is used to name the virtual machine in the `/vcenter` hierarchy. When the type is Host, this parameter is the host name or the IP address of the host that is added. The host name must be a fully qualified domain name (FQDN). When the type is PerfCollector, this parameter is a user-friendly name for the performance collector. **Note:** Any `%` (percent) character used in this name parameter must be escaped, unless it is used to start an escape sequence. Clients may also escape any other characters in this name parameter.

Create		
Name	**Type**	**Description**
type	xsd:string	Type of the new object. Type is one of the following: • Container • Farm • VirtualMachineGroup • VirtualMachine • TaskSchedule • Host • PerfCollector
initial	xsd:anyType	(Optional) XML document providing additional information to create the object: • XML document of type VirtualMachineSpec is passed to create a VirtualMachine. The initial.host field indicates the destination host on which the virtual machine will be created. • XML document of type HostSpec is passed to create a Host. • XML document of type TaskScheduleSpec is passed to create a task. • XML document of type PerfCollector is passed to create a PerfCollector. This document is a specification for the new performance collector, including the parent collector, filter, and number of samples, and optionally, a different sampling interval than the default. **Note:** If clients leave the spec field uninitialized (or NULL), then all performance statistics from the source are returned • Omitted for Container, Farm, and VirtualMachineGroup.

Creating a New Task

Clients may use the create operation to create a new task that is scheduled to run at some point in the future. Only the following operations can be scheduled:

- Power operations — StopVM, StartVM, and ResetVM.
- PutUpdates — changing the resource settings of a virtual machine. Clients must pass a handle (and not a vHandle) for a scheduled PutUpdates operation. See PutUpdates on page 146 for more information.
- MigrateVM
- MoveVM
- CloneVM

- CreateTemplate

For detailed information on creating a scheduled task, refer to the task scheduling and monitoring section in the *Virtual Infrastructure SDK Programming Guide*. Here, we provide a summary of the steps required to schedule a suspend operation using StopVM as an example.

This example describes creating a task schedule for a suspend operation (using the StopVM operation) on a virtual machine.

1. Instantiate the TaskScheduleSpec stub class.
2. Fill in the KeyedValue array with the arguments for StopVM: the handle to the virtual machine, the suspend flag and the soft flag. See StopVM on page 135 for more information on these parameters.
3. Determine the name for the task schedule (name parameter), the operationName (StopVM), and the recurrence of the schedule. See TaskScheduleSpec on page 90.
4. Pass the TaskScheduleSpec to the Create operation.

Note: If you are scheduling a resume operation for StartVM, then you need to add an additional parameter to the KeyedValue array where the key is "resume" and the val is "true".

Adding a Host

When clients use the Create operation to add a host to the Web service inventory, the user name and password fields in HostSpec are optional.

- If no user name and password are supplied, then the host is added to the Web service inventory in the Disabled state and the virtual machines in the host are not visible to the client. The client may then use the EnableHost operation to change the host state to Enabled.
- If an incorrect user name or password is supplied, then the host is not added to the inventory. Instead, the Web service adds the host in the Disabled state. However, the host does not have any virtual machines because the Web service is unable to determine what virtual machines are present until the host is Enabled.
- Providing an empty user name or providing a password without a corresponding user name is not allowed. However, providing a valid user name with either no password or an empty password, is allowed.

Creating a Virtual Machine

Clients can use the Create operation to create a new virtual machine. However, to create the virtual disk itself, use the CreateVirtualDisk operation. (See CreateVirtualDisk on page 156.)

When using the Create operation, the initial parameter is an XML document of type VirtualMachineSpec describing the virtual machine. (See VirtualMachineSpec on page 104.) This VirtualMachineSpec datatype may include the host name, the guest operating system, the

location of the virtual machine configuration file, and the virtual hardware. (See VirtualHardware on page 97.)

Note: Clients cannot specify two or more devices, of the same type (for example, two NICs or two floppy drives) in the hardware specification when creating a virtual machine.

Instead, during the initial Create operation, specify one of the devices (for example, one NIC) in the hardware specification, then specify the additional device (second NIC) by using a PutUpdates call on that newly created virtual machine.

Creating a Perf Collector

A perf collector is an object that collects a certain set of statistics at a specified interval frequency. There are two types of performance collectors:

- VirtualCenter perf collectors — VirtualCenter perf collectors specify a sampling interval and are visible in the VMware VirtualCenter application.
- Filtered perf collector — Filtered perf collectors (children) are filtered from an existing VirtualCenter perf collector (parent) and are not visible in the VirtualCenter application.

Clients can create both types of performance collectors by using the Create operation.

- A new VirtualCenter perf collector with a different sampling interval from any other VirtualCenter perf collectors.

  Specify only the name and the sampling interval (initial parameter comprising XML document of type PerfCollector).

- A perf collector that filters the statistics of an existing VirtualCenter perf collector.

  Specify the name and the filter (initial parameter comprising XML document of type PerfCollector). This XML document *must include* the handle to the parent VirtualCenter perf collector. The filter should also include the source (host or virtual machine being queried), the samples, and the performance statistics of interest.

  Note: Do not specify the sampling interval. The filtered perf collector has the same sampling interval as its parent VirtualCenter perf collector.

  Note: If clients leave the spec field uninitialized (or NULL), then all performance statistics from the source are returned.

Output Message — CreateResponseMsg

CreateResponse		
Name	**Type**	**Description**
returnval	xsd:string	Upon success, the handle to the newly created object is returned.

Fault

FaultInfo		
Name	**Type**	**Description**
kind	FaultKind	• AlreadyExists — The target name is already defined. • BadRequest — The request is not well-formed. • HandleFault – The handle, passed as a parameter, has encountered errors. • NotApplicable — One of the specified objects is of the wrong type; for example, a host handle does not refer to a host. • OperationFailed — The server could not successfully complete this operation.
code	xsd:int	Integer, which gives the error code that is classified in a manner similar to HTTP error codes.
info	xsd:string	See the FaultInfo datatype for details.
val	xsd:anyType	See the FaultInfo datatype for details.

Example One

This example illustrates creating a new virtual machine in `/vm` directory.

Parameter	Description
Input parameter	• handle — Handle for `/vm`. • name — Name of the virtual machine as it should appear in the `/vcenter` hierarchy. • type — VirtualMachine. • initial — XML document of type VirtualMachineSpec describing the virtual machine. The initial.host field indicates the destination host on which the virtual machine will be created.
Output parameter	• resultName — Handle to the newly created virtual machine. You may use this handle to get additional information or monitor the virtual machine. The virtual machine also appears in the `/vcenter` Farm, to which the host belongs.

The type.host field must be specified in the virtual machine XML document. This field indicates the destination host on which the virtual machine is created. The virtual machine is created in the `/vcenter` Farm, to which the host belongs.

Example Two

This example illustrates creating a new virtual machine in the `/vcenter` directory.

Parameter	Description
Input parameter	• handle — Handle to`/vcenter/Container/Farm1.` • name — name of the virtual machine. • type — VirtualMachine. • initial — XML document of type VirtualMachineSpec describing the virtual machine. The initial.host field indicates the destination host on which the virtual machine will be created. The destination host must be a member of the Farm that is specified in the handle parameter.

Parameter	Description
Output parameter	• resultName — Handle to `/vcenter/Container/Farm1/name`, that is the newly created virtual machine. The virtual machine also appears in the `/vm` hierarchy, with the assigned UUID for this virtual machine.

CreateVirtualDisk

The CreateVirtualDisk operation requests the creation of a virtual disk for a virtual machine. Clients can determine whether the virtual disk creation is successful, by separately monitoring the task that was returned.

Note: Clients cannot use the Create operation to create a new virtual disk.

Note: Virtual disks are not independent entities but are always associated with a virtual machine. The CreateVirtualDisk operation requires a virtual machine handle as one of its input parameters, therefore the virtual machine must also exist before this operation can be called. The diskInfo parameter describes the virtual disk that will be created.

Clients can use the CreateVirtualDisk operation to create disks that are raw disk mappings to a raw disk (physical disk) volume. Specify a DiskDeviceInfo structure and a RawDisk volume in the VirtualDiskInfo structure. See DiskDeviceInfo on page 51, DiskKind on page 53, andVirtualDiskInfo on page 94.

Permissions

The client must have Configure rights on the virtual machine.

Input Message — CreateVirtualDiskRequestMsg

CreateVirtualDisk		
Name	**Type**	**Description**
vm	xsd:string	Handle specifying the target virtual machine.
diskInfo	VirtualDiskInfo	Specifies the parameters that control the creation of a virtual disk.

Note: If a client attempts to create a virtual disk that already exists, the operation returns a successful task. It does not fail and does not return a fault. However, another virtual disk is not created.

Note: If the client specifies a name for the virtual disk in the CreateVirtualDiskRequestMsg, it is ignored in this release.

Output Message — CreateVirtualDiskResponseMsg

Once the operation has been initiated, the request returns a task handle to the client. The client may monitor the task for the progress of the operation.

CreateVirtualDiskResponse		
Name	**Type**	**Description**
returnval	ViewContents	The response specifies the ViewContents of the task performing the CreateVirtualDisk operation. • ViewContents.handle – Handle for the task object. • ViewContents.vHandle – Versioned handle for the task object. • ViewContents.body – XML document of type Task that describes the task object.

Fault

FaultInfo		
Name	**Type**	**Description**
kind	FaultKind	• BadRequest — The request is not well-formed. • HandleFault – The handle, passed as a parameter, has encountered errors. • NotApplicable — The specified handle is not a virtual machine. • OperationFailed — The server could not successfully complete this operation.
code	xsd:int	Integer, which gives the error code that is classified in a manner similar to HTTP error codes.
info	xsd:string	See the FaultInfo datatype for details.
val	xsd:anyType	See the FaultInfo datatype for details.

Rename

The Rename operation requests a change to the name of an existing object and optionally moves it within the `/vcenter` hierarchy.

Note: You may not use the Rename operation to migrate a virtual machine.

The following objects types may be renamed:

- Container
- Farm
- VirtualMachineGroup
- VirtualMachine

Note: You can only move a virtual machine or VirtualMachineGroup to a new VirtualMachineGroup. Similarly, you can only move a Farm or a FarmGroup (Container) to a new FarmGroup. You cannot move a VirtualMachineGroup to a new Farm and you cannot move objects across Farms.

Permissions

The client must have Browse and Configure rights for both the current and new Container for the object.

Input Message — RenameRequestMsg

Performing the Rename operation on an object, without moving the object, causes updates on the parent object. A Rename operation that makes changes to the `/vcenter` hierarchy causes updates on the parent and destination objects. The existing handle and vHandles to the renamed object continue to work.

Rename		
Name	**Type**	**Description**
handle	xsd:string	Handle of the existing object.
destHandle	xsd:string	(Optional) This handle is an optional parameter that specifies the destination for the object being moved. If this parameter is not specified, then the object is not moved, but is simply renamed in its current location.

Rename		
Name	**Type**	**Description**
name	xsd:string	New name for the object. If you do not want a name change (it is only a move), then pass in the current name of the object. **Note:** Any `%` (percent) character used in this name parameter must be escaped, unless it is used to start an escape sequence. Clients may also escape any other characters in this name parameter.

The following rules apply to the parameters:

- The destination object, pointed to by destHandle, must be capable of holding the type of object specified by the handle parameter. For example, if handle refers to a virtual machine, then destHandle must refer to an object of type Farm or VirtualMachineGroup. destHandle cannot refer to an object of type Container, as Container objects cannot hold virtual machines.
- If the handle refers to a VirtualMachine, then it cannot be moved across Farms in the `/vcenter` hierarchy. You must use the MigrateVM operation to move virtual machines from one host to another.

Output Message — RenameResponseMsg

Upon success, an empty response message is returned.

Fault

FaultInfo		
Name	**Type**	**Description**
kind	FaultKind	• AlreadyExists — The target name already refers to an object. • BadRequest — The request is not well-formed. • HandleFault – The handle, passed as a parameter, has encountered errors. • NotApplicable — The specified handle does not refer to an existing object. • OperationFailed — The server could not successfully complete this operation.
code	xsd:int	Integer, which gives the error code that is classified in a manner similar to HTTP error codes.

FaultInfo		
Name	**Type**	**Description**
info	xsd:string	See the FaultInfo datatype for details.
val	xsd:anyType	See the FaultInfo datatype for details.

Example One

This example illustrates renaming a virtual machine from Windows2000 to Win2K:

- handle —Handle for `/vcenter/Farm/VirtualMachineGroup/Windows2000`.
- destHandle — Omitted.
- name — `Win2K`.

The client receives the following updates (if it has requested GetUpdates):

- `ChangeOp == edit` — For the vHandle for `/vcenter/Farm/VirtualMachineGroup/`.
- `ChangeOp == edit` — For the vHandle for the virtual machine Windows2000, since the name of the virtual machine changed.
- `ChangeOp == edit` — For the path to the virtual machine.

Example Two

This example illustrates moving a virtual machine in the `/vcenter` hierarchy. Note that you may not move virtual machines from one Farm to another Farm.

- handle —Handle for `/vcenter/Farm/VirtualMachineGroup/Windows2000`.
- destHandle — Handle for `/vcenter/Farm/`.
- name — `Windows2000` (current name of the virtual machine since no name change is desired).

The client receives the following updates (if it has requested GetUpdates):

- `ChangeOp == move` — For the vHandle for `/vcenter/Farm/VirtualMachineGroup/` and `/vcenter/Farm/`.
- `ChangeOp == edit` — For the vHandle for the virtual machine Windows2000 since the path in the hierarchy changed for the virtual machine.

Example Three

This example illustrates renaming a Virtual Machine Group in the `/vcenter` hierarchy.

- handle — Handle for `/vcenter/Farm/VirtualMachineGroup`.
- destHandle — Omitted.
- name — `VMG`.

The client receives the following updates (if it has requested GetUpdates):

- `ChangeOp == edit` — For the vHandle for `/vcenter/Farm/`.
- `ChangeOp == edit` — For the vHandles for virtual machines in the VirtualMachineGroup since the path in the hierarchy changed for the virtual machines.
- `ChangeOp == edit` — For the path to the virtual machine.

Example Four

This example illustrates moving a virtual machine in the `/vcenter` hierarchy and renaming it. Note that you may not move virtual machines from one Farm to another Farm.

- handle — Handle for `/vcenter/Farm/VirtualMachineGroup/Windows2000`.
- destHandle — Handle for `/vcenter/Farm/`.
- name — `Win2K`.

The client receives the following updates (if it has requested GetUpdates):

- `ChangeOp == move` — For the vHandle for `/vcenter/Farm/VirtualMachineGroup` and `/vcenter/Farm`.
- `ChangeOp == edit` — For the vHandle for the virtual machine Windows2000 since the path in the hierarchy and the name of the virtual machine changed.

Delete

The Delete operation requests that an object be deleted. This operation may release all storage for an object. In particular, deleting a virtual machine deletes all the virtual disk files associated with that virtual machine. Deleting a host only removes it from the Web service and VirtualCenter. The virtual machines on the deleted host are not deleted, but cannot be seen by the client until the host is added back to the Web service (or VirtualCenter).

Note: If the client deletes a host, then migrationEnabled is turned off in the HostInfo datatype.

Clients may delete Container, Farm, VirtualMachineGroup, VirtualMachine, and Host objects. These objects are not required to be empty.

Note: Creating and deleting hosts in quick succession may sometimes fail.

Note: When a client deletes a top-level object (that contains other objects), then a delete change is only shown for the top-level object. For example, if a client deletes a Farm, then the client sees a delete change on the Farm, but not on any hosts in the Farm, or any virtual machines on the hosts.

A virtual machine must be powered off (stopped) or suspended, and cannot be migrating, for the delete operation to succeed. If a client wants to delete a virtual machine but not its virtual disk, then the client must do the following steps:

1. Stop the virtual machine.
2. Using the PutUpdates operation, disconnect the virtual disk(s) of the virtual machine.
3. Delete the virtual machine.

Permissions

The client must have Configure rights on the object and on the object's Container.

Input Message — DeleteRequestMsg

Delete		
Name	**Type**	**Description**
handle	xsd:string	Handle to the object that will be deleted.

Output Message — DeleteResponseMsg

Upon success, an empty response message is returned.

Fault

FaultInfo		
Name	**Type**	**Description**
kind	FaultKind	• BadRequest — The request is not well-formed. • HandleFault – The handle, passed as a parameter, has encountered errors. • OperationFailed — The server could not successfully complete this operation.
code	xsd:int	Integer, which gives the error code that is classified in a manner similar to HTTP error codes.
info	xsd:string	See the FaultInfo datatype for details.
val	xsd:anyType	See the FaultInfo datatype for details.

QueryPerfData

The QueryPerfData operation requests either the most recent (current) data, or historical performance data from an existing performance collector. You use arguments to specify the start time, the number of samples, and the filtering of the data. We define filtering as the specification of the data of interest to the client, and the granularity of the data.

However, the QueryPerfData operation cannot guarantee that the returned data completely matches the request. The start time, the number of samples, the granularity, and the returned data may be different than the request. Also, there may be "holes" in the data when the data is unavailable. If the operation cannot return performance data at the requested date or granularity, it attempts to return whatever data it can, beginning at the requested start time.

Permissions

The client must have Browse rights on the virtual machines and hosts, whose statistics are being queried.

Input Message — QueryPerfDataRequestMsg

QueryPerfData		
Name	**Type**	**Description**
handle	xsd:string	Handle to the performance collector that is being queried.
startTime	xsd:dateTime	(Optional) Start time of the samples to be returned. If startTime is omitted, then the current time (or the latest samples) is assumed.
samples	xsd:int	Number of sample in the request.
filter	PerfFilter	(Optional) If desired, the filter can be used to specify the granularity, or the level of detail in the response. **Note:** Do not use the filter parameter when querying a filtered perf collector or the operation returns an error.

Output Message — QueryPerfDataResponseMsg

QueryPerfDataResponse		
Name	**Type**	**Description**
returnval	PerfCollector	Specifies the filter applied to the request and the performance statistics from the various sources. For more information, see PerfCollector on page 82.

Note: If you see output similar to `/vpx/vm/#000512` or `/vpx/host/#000a376` for the PerfSourceType "key" field in the PerfCollection object, then the specified virtual machine or host has been deleted.

Fault

FaultInfo		
Name	**Type**	**Description**
kind	FaultKind	• BadRequest — The request is not well-formed. • HandleFault – The handle, passed as a parameter, has encountered errors. • NotApplicable — The specified source is not a host or a virtual machine. • OperationFailed — The server could not successfully complete this operation.
code	xsd:int	Integer, which gives the error code that is classified in a manner similar to HTTP error codes.
info	xsd:string	See the FaultInfo datatype for details.
val	xsd:anyType	See the FaultInfo datatype for details.

QueryPerfData2

The QueryPerfData2 operation is designed for getting the extra statistics contained in the new CPUPerf2, MemoryPerf2, and VirtualMachineMemoryPerf2 datatypes. Its use and description is exactly the same as the QueryPerfData operation, described in the preceding section. However, QueryPerfData2 returns the CPUPerf2, MemoryPerf2, and VirtualMachineMemoryPerf2 data types whereas QueryPerfData returns the CPUPerf, MemoryPerf, and VirtualMachineMemoryPerf datatypes.

Note: Use the QueryPerfData2 operation to obtain the extra statistics contained in these new datatypes. Clients cannot use any other operation to obtain these statistics.

Permissions

The client must have Browse rights on the virtual machines and hosts, whose statistics are being queried.

Input Message — QueryPerfData2RequestMsg

QueryPerfData2		
Name	**Type**	**Description**
handle	xsd:string	Handle to the performance collector that is being queried.
startTime	xsd:dateTime	(Optional) Start time of the samples to be returned. If startTime is omitted, then the current time (or the latest samples) is assumed.
samples	xsd:int	Number of sample in the request.
filter	PerfFilter	(Optional) If desired, the filter can be used to specify the granularity, or the level of detail in the response. **Note:** Do not use the filter parameter when querying a filtered perf collector or the operation returns an error.

Output Message — QueryPerfData2ResponseMsg

QueryPerfData2Response		
Name	**Type**	**Description**
returnval	PerfCollector	Specifies the filter applied to the request and the performance statistics from the various sources. For more information, see PerfCollector on page 82.

Note: If you see output similar to `/vpx/vm/#000512` or `/vpx/host/#000a376` for the PerfSourceType "key" field in the PerfCollection object, then the specified virtual machine or host has been deleted.

Fault

FaultInfo		
Name	**Type**	**Description**
kind	FaultKind	• BadRequest — The request is not well-formed. • HandleFault – The handle, passed as a parameter, has encountered errors. • NotApplicable — The specified source is not a host or a virtual machine. • OperationFailed — The server could not successfully complete this operation.
code	xsd:int	Integer, which gives the error code that is classified in a manner similar to HTTP error codes.
info	xsd:string	See the FaultInfo datatype for details.
val	xsd:anyType	See the FaultInfo datatype for details.

ChangePermissions

The ChangePermissions operation modifies or deletes permissions on an object, such as a virtual machine, host or Container.

Permissions

The client must have Administer rights on the vHandle.

Input Message — ChangePermissionsRequestMsg

ChangePermissions		
Name	**Type**	**Description**
vHandle	xsd:string	Versioned handle to the object for which the permissions will be changed.
permissions	PermissionList	List of Permissions. A Permission comprises a user string and a rights string.

Output Message — ChangePermissionsResponseMsg

Upon success, an empty response message is returned. A GetUpdates request on the parent handle of the object returns the new permissions for the object.

Fault

FaultInfo		
Name	**Type**	**Description**
kind	FaultKind	• BadRequest — The request is not well-formed. • HandleFault – The handle, passed as a parameter, has encountered errors. • NotApplicable — The specified source is not a host or a virtual machine. • OperationFailed — The server could not successfully complete this operation. • PermissionDenied — The client does not have sufficient permissions to perform the operation.
code	xsd:int	Integer, which gives the error code that is classified in a manner similar to HTTP error codes.
info	xsd:string	See the FaultInfo datatype for details.

FaultInfo		
Name	**Type**	**Description**
val	xsd:anyType	See the FaultInfo datatype for details.

SnapshotVM

The SnapshotVM operation takes a picture of a virtual machine at a particular point in time. This snapshot maintains the state of the virtual disks.

- ESX Server 2.5 — The virtual machine must be powered on. The client takes a non-memory snapshot that includes data from all virtual disks, except disks in nonpersistent mode. However, the snapshot does not include the contents of the virtual machine's memory.

  If the virtual machine has no redo logs, then new redo logs are added. However, if the virtual machine already has redo logs, then regardless of the disk mode, the redo logs are committed to the base virtual disk(s) when the snapshot occurs and new redo logs are added.

  Note: If you perform the SnapshotVM operation on a virtual machine with redo logs and virtual disks in undoable mode, then the existing redo logs are committed to the base virtual disks (cannot be undone) before new redo logs are added.

- GSX Server 3.1 — The client takes a memory snapshot that includes the virtual disk data, the virtual machine's settings, and the contents of the virtual machine's memory.

  If the virtual machine has no existing snapshot, then a new snapshot is created. However, if the virtual machine already has an existing snapshot, then the existing snapshot is committed to the base virtual disk(s) and a new snapshot is taken.

  Note: The SnapshotVM operation on a powered-off virtual machine on a GSX Server host can fail even though the operation appears to complete successfully. For GSX Server, we recommend that clients perform these operations only on powered-on virtual machines only through the Web service interface.

Permissions

The client must have Configure rights on the virtual machine.

Input Message — SnapshotVMRequestMsg

SnapshotVM		
Name	**Type**	**Description**
vm	xsd:string	Handle specifying the target virtual machine.

Output Message — SnapshotVMResponseMsg

Once the operation has been initiated, the request returns a task handle to the client. The client may monitor the task for the progress of the operation.

SnapshotVMResponse		
Name	**Type**	**Description**
returnval	ViewContents	The response specifies the ViewContents of the task performing the SnapshotVM operation: • ViewContents.handle — Handle for the task object. • ViewContents.vHandle — Versioned handle for the task object. • ViewContents.body — XML document of type Task that describes the task object.

Fault

FaultInfo		
Name	**Type**	**Description**
kind	FaultKind	• BadState — The operation cannot be performed, because the virtual machine is in an incorrect state. • OperationFailed — The server could not successfully complete this operation.
code	xsd:int	Integer, which gives the error code that is classified in a manner similar to HTTP error codes.
info	xsd:string	See the FaultInfo datatype for details.
val	xsd:anyType	See the FaultInfo datatype for details.

ConsolidateVM

The ConsolidateVM operation commits the changes to the base virtual disk(s).

- ESX Server 2.5 — The virtual machine must be powered on. Any outstanding redo logs are committed.
- GSX Server 3.1 — The virtual machine must be powered off. The snapshot is committed.

Permissions

The client must have Configure rights on the virtual machine.

Input Message — ConsolidateVMRequestMsg

ConsolidateVM		
Name	**Type**	**Description**
vm	xsd:string	Handle specifying the target virtual machine.

Output Message — ConsolidateVMResponseMsg

Once the operation has been initiated, the request returns a task handle to the client. The client may monitor the task for the progress of the operation.

ConsolidateVMResponse		
Name	**Type**	**Description**
returnval	ViewContents	The response specifies the ViewContents of the task performing the ConsolidateVM operation: • ViewContents.handle — Handle for the task object. • ViewContents.vHandle — Versioned handle for the task object. • ViewContents.body — XML document of type Task that describes the task object.

Fault

FaultInfo		
Name	**Type**	**Description**
kind	FaultKind	• BadState — The operation cannot be performed, because the virtual machine is in an incorrect state. • OperationFailed — The server could not successfully complete this operation.
code	xsd:int	Integer, which gives the error code that is classified in a manner similar to HTTP error codes.
info	xsd:string	See the FaultInfo datatype for details.
val	xsd:anyType	See the FaultInfo datatype for details.

RevertVM

The RevertVM operation discards any existing snapshot and returns the virtual machine to its state preceding the snapshot. In this release, the RevertVM operation is only supported on GSX Server.

Note: Trying to revert a snapshot, immediately after it has been created by the SnapshotVM operation, may sometimes fail. This is a host limitation and a small delay is needed before initiating the RevertVM operation.

Note: The RevertVM operation on a powered-off virtual machine on a GSX Server host can fail even though the operation appears to complete successfully. For GSX Server, we recommend that clients perform these operations only on powered-on virtual machines through the Web service interface.

Permissions

The client must have Configure rights on the virtual machine.

Input Message — RevertVMRequestMsg

RevertVM		
Name	**Type**	**Description**
vm	xsd:string	Handle specifying the target virtual machine.

Output Message — RevertVMResponseMsg

Once the operation has been initiated, the request returns a task handle to the client. The client may monitor the task for the progress of the operation.

RevertVMResponse		
Name	**Type**	**Description**
returnval	ViewContents	The response specifies the ViewContents of the task performing the RevertVM operation: • ViewContents.handle — Handle for the task object. • ViewContents.vHandle — Versioned handle for the task object. • ViewContents.body — XML document of type Task that describes the task object.

Fault

FaultInfo		
Name	**Type**	**Description**
kind	FaultKind	• BadState — The operation cannot be performed, because the virtual machine is in an incorrect state. • OperationFailed — The server could not successfully complete this operation.
code	xsd:int	Integer, which gives the error code that is classified in a manner similar to HTTP error codes.
info	xsd:string	See the FaultInfo datatype for details.
val	xsd:anyType	See the FaultInfo datatype for details.

CHAPTER 6

Virtual Computing Operations

This chapter describes the following virtual computing operations:

- EnableHost on page 179
- DisableHost on page 181
- MoveVM on page 182
- MigrateVM on page 185
- CloneVM on page 187
- CreateTemplate on page 190
- RunTask on page 192
- EndTask on page 193

Performing Virtual Computing Operations

The Virtual Computing operations enable a higher-level functionality for managing virtual machines across a farm of physical resources. These operations enable the client program to perform the following functions:

Virtual Machine Migration, including VMotion

Clients may invoke a virtual machine migration, that does all of the standard pre-migration verifications. All of the historical data for each virtual machine is preserved, including the unique virtual machine identifier. This prevents any external databases that are dependent on such information, from being disrupted.

Note: VMotion is not supported for GSX Server in this release.

Virtual Machines Are Managed Independently of Hosts

Clients may refer to the virtual machine as an independent entity. Any operations performed on an existing virtual machine may be invoked without reference to physical hosts. There are two exceptions: provisioning a new virtual machine or migrating a virtual machine. In both cases, you must specify a target host.

Virtual Machine Provisioning and Deployment

The virtual machine provisioning capability includes the ability to do the following:

- Select a template from the repository
- Customize the images prior to deployment
- Deploy the template with its customizations to a target host over the network, or use a SAN-based copy.

Task Scheduling

You may schedule all operations performed through the Web service interface to occur at a particular time, or to recur at a particular date and time.

EnableHost

When clients use the Create operation (with a valid user name and password) to add a host to the Web service inventory, the host is automatically enabled for virtual machine operations. The client may use the DisableHost operation to disable virtual machine operations, but still keep the host under management.

A host in the Disabled state may be enabled using the EnableHost operation. When a host is in the Enabled state, new virtual machines may be discovered. These virtual machines are listed in the `/vm` hierarchy, and also in the `/vcenter` hierarchy, under a special VirtualMachineGroup called "DiscoveredVMs" that appears in the Farm where the specified host resides.

If a host is enabled that has a running virtual machine with a missing or duplicate UUID, then the virtual machine is placed in a special directory, `/unknownVM`. The Web service assigns a small serial number to the virtual machine until it receives a UUID. As soon as the virtual machine is powered off, VirtualCenter assigns a valid UUID and the Web service removes the virtual machine from `/unknownVM` and places it in `/vm`. When this happens, the handle for `/unknownVM/#` (which is the same for `/vcenter/<path>/VM`) is valid and will continue to work.

Permissions

The client must have Configure rights on the Farm.

Input Message — EnableHostRequestMsg

EnableHost		
Name	**Type**	**Description**
host	xsd:string	Handle specifying the target host.
userName	xsd:string	(Optional) User name that VirtualCenter uses to connect to the host specified by the handle.
password	xsd:string	(Optional) Password that VirtualCenter uses to connect to the host specified by the handle.

When a host is enabled, its virtual machines are shown in their current state. However, if VirtualCenter loses network connection to a host, then it will put a host into the Disabled state. The host's virtual machines remain in their current state, with the detail field indicating that the virtual machines are disconnected.

Output Message — EnableHostResponseMsg

Upon success, an empty response message is returned.

Fault

FaultInfo		
Name	**Type**	**Description**
kind	FaultKind	• BadRequest — The request is not well-formed. • HandleFault – The handle, passed as a parameter, has encountered errors. • OperationFailed — The server could not successfully complete this operation.
code	xsd:int	Integer, that gives the error code that is similarly classified to HTTP error codes.
info	xsd:string	See the FaultInfo datatype for details.
val	xsd:anyType	See the FaultInfo datatype for details.

DisableHost

Clients may add a host in the Disabled state to the Web service inventory by using the Create operation, without supplying the user name and password. The host may also be disabled from virtual machine operations by using the DisableHost operation.

The DisableHost operation does not remove the host name from the Farm. You must perform a Delete operation to remove the host. We keep these two operations separate, so that you can keep hosts under management, without having to remove them.

Permissions

The client must have Configure rights on the Farm.

Input Message — DisableHostRequestMsg

DisableHost		
Name	**Type**	**Description**
host	xsd:string	Handle specifying the target host.

When a host is disabled, the virtual machines remain in their current state, with additional detail indicating that the virtual machines are disconnected.

Output Message — DisableHostResponseMsg

Upon success, an empty response message is returned.

Fault

FaultInfo		
Name	**Type**	**Description**
kind	FaultKind	• BadRequest — The request is not well-formed. • HandleFault – The handle, passed as a parameter, has encountered errors. • OperationFailed — The server could not successfully complete this operation.
code	xsd:int	Integer, that gives the error code that is similarly classified to HTTP error codes.
info	xsd:string	See the FaultInfo datatype for details.
val	xsd:anyType	See the FaultInfo datatype for details.

MoveVM

The MoveVM operation moves a virtual machine's virtual disk(s) to a different location. You may (optionally) also use this operation to move the virtual machine to a different host. The virtual machine must be powered off. You cannot perform this operation on a virtual machine while it is in the poweredOn or suspended states.

These are the restrictions for the MoveVM operation in the current release:

- host input parameter must always be specified.
- dataLocator input parameter is ignored.
- disk input parameter may specify zero or one VirtualDiskDestination.
  - If zero (not specified), the Web service picks the default (first) datastore from the host.

    If the virtual machine is being moved to a different host, and both hosts have only one (in this case, shared) datastore, then the virtual disk is not moved.

    If the destination host has two datastores, then the Web service attempts to get the first (default) datastore from the destination host and specifies that datastore to VirtualCenter, as the destination for the virtual disk(s). If the first datastore is different from the shared datastore, then the virtual disk(s) are copied to the first datastore, since the Web service explicitly specified it. However, if the first datastore is the same as the shared datastore, then the virtual disk(s) is not moved.
  - If one (specified), disk.dataLocator must be a datastore accessible from the destination host.

    If the virtual machine was created on a datastore shared by the source and destination hosts, then the virtual disk(s) is not moved if the client explicitly specifies the destination datastore as the "shared" datastore.

  Note: The datastore is used as the destination for all of the virtual machine's virtual disks.

Permissions

The client must have Configure rights on the virtual machine and on the Farm of its host. If you are also planning to move this virtual machine to a different host, then the client must also have Configure rights on the target (destination) host.

Input Message — MoveVMRequestMsg

MoveVM		
Name	**Type**	**Description**
vm	xsd:string	Handle to the virtual machine.
host	xsd:string	(Required for this release) Handle to the destination host. If this parameter is specified, then both the virtual machine and its virtual disk(s) are moved. If this parameter is omitted, only the virtual disk(s), and not the virtual machine, is moved.
dataLocator	xsd:string	(Optional) Path describing the location of the virtual machine configuration file. If this parameter is omitted, the Web service determines the location of this configuration file. **Note:** The dataLocator parameter is ignored in this release.
disk	VirtualDiskDestination[]	(Optional) Specifies the destination for all disks in the virtual machine. If this parameter is omitted, the Web service determines the destination location of the virtual disk(s). If specified, each virtual disk contains two fields: • key — Key of the virtual disk; for example, `#_scsiDev0:0` or `#ide_Dev1:0`. The key of the VirtualDiskDestination is ignored in this release. • dataLocator — Path describing the destination datastore for the virtual disk file(s).

Output Message — MoveVMResponseMsg

The request returns once the operation has been initiated by returning a task handle back to the client. The client may monitor the task for progress of the operation. Upon success, the virtual

disk(s) has been moved to the specified destination and the virtual machine has been moved to the destination host (if a destination host was specified).

MoveVMResponse		
Name	**Type**	**Description**
returnval	ViewContents	The response specifies the ViewContents of the task performing the MoveVM operation. • ViewContents.handle – Handle for the task object. • ViewContents.vHandle – Versioned handle for the task object. • ViewContents.body – XML document of type Task that describes the task object.

Fault

Upon failure, the original configuration of the virtual machine and its virtual disk(s) are retained.

FaultInfo		
Name	**Type**	**Description**
kind	FaultKind	• BadRequest — The request is not well-formed. • BadState — The virtual machine is powered on or suspended. • HandleFault – The handle, passed as a parameter, has encountered errors. • NotApplicable — The specified handle does not refer to a virtual machine. • OperationFailed — The server could not successfully complete this operation.
code	xsd:int	Integer, that gives the error code that is similarly classified to HTTP error codes.
info	xsd:string	See the FaultInfo datatype for details.
val	xsd:anyType	See the FaultInfo datatype for details.

MigrateVM

The MigrateVM operation starts the process of migrating a virtual machine to a specific host, without moving the virtual disk(s) from its current location. In this release, the virtual machine must be in the poweredOn state for the MigrateVM operation to succeed.

If you want to move a virtual machine's virtual disk(s), use the MoveVM operation. The disks must be on a SAN shared by both the source and target hosts. For more information, see MoveVM on page 182.

Note: The MigrateVM operation is not supported for a GSX Server host.

Permissions

The client must have Configure rights on the virtual machine and on the Farm.

Input Message — MigrateVMRequestMsg

MigrateVM		
Name	**Type**	**Description**
vm	xsd:string	Handle to the virtual machine. The virtual machine must be either in the poweredOff or poweredOn states, for migration to start.
host	xsd:string	Handle to the destination host.
priority	Level	(Optional) Determines whether resources are preallocated before migration starts. Takes one of the following values: • low • normal • high The default value is "high".
dataLocator	xsd:string	Optional) Path describing the location of the virtual machine configuration file. If this parameter is omitted, the Web service determines the location of this configuration file. **Note:** The dataLocator parameter is ignored in this release.

Output Message — MigrateVMResponseMsg

The request returns once the operation has been initiated by returning a task handle back to the client. You can determine whether the migration is successful, by separately monitoring the task that is performing the migration or by monitoring the virtual machine state (specifically the detail

or host fields in VirtualMachineState). If the virtual machine has been successfully migrated, then the host field should contain the target host name.

MigrateVMResponse		
Name	**Type**	**Description**
returnval	ViewContents	The response specifies the ViewContents of the task performing the MigrateVM operation. • ViewContents.handle – Handle for the task object. • ViewContents.vHandle – Versioned handle for the task object. • ViewContents.body – XML document of type Task that describes the task object.

Fault

FaultInfo		
Name	**Type**	**Description**
kind	FaultKind	• BadRequest — The request is not well-formed. • BadState — The virtual machine is suspended or powered off. • HandleFault – The handle, passed as a parameter, has encountered errors. • NotApplicable — The specified handle does not refer to a virtual machine. • OperationFailed — The server could not successfully complete this operation.
code	xsd:int	Integer, that gives the error code that is similarly classified to HTTP error codes.
info	xsd:string	See the FaultInfo datatype for details.
val	xsd:anyType	See the FaultInfo datatype for details.

CloneVM

The CloneVM operation creates a new virtual machine by using as its source, an existing virtual machine or a template. Clients can choose to customize a cloned virtual machine by populating the info field (described in the following table).

Virtual machines must be powered off in order for the CloneVM operation to succeed. If the source is a template located on a host's datastore, then the newly cloned virtual machine must also reside on the same host where the datastore is located. However, if hosts are on the same SAN and share the same datastore, then the CloneVM operation (from a template) may be done across these hosts.

Clients can also configure the guest operating system as part of the CloneVM operation by passing in this information from generated stub classes. Refer to the *Virtual Infrastructure SDK Programming Guide* for additional information.

Permissions

The client must have Configure rights on the virtual machine and on the Farm of its host. If you are also planning to clone this virtual machine to a different host, then the client must also have Configure rights on the target (destination) host.

Input Message — CloneVMRequestMsg

CloneVM		
Name	**Type**	**Description**
srcHandle	xsd:string	Handle to the (source) virtual machine or template that will be cloned. Clients cannot clone a running virtual machine; the virtual machine must be powered off. Templates stored in the VMware VirtualCenter Server Repository datastore can be cloned to any host. if a template is stored on a different datastore, then clients can only deploy the template to a host that can access the template's datastore.
parentHandle	xsd:string	Handle to the destination Farm or virtual machine group in which the cloned virtual machine will be created.
destHostHandle	xsd:string	Handle to the host in which the cloned virtual machine will reside.
name	xsd:string	Name of the newly cloned virtual machine.
datastore	xsd:string	Location on the destination host where the cloned virtual machine's configuration files and virtual disks will reside.

CloneVM		
Name	**Type**	**Description**
customization	xsd:anyType	(Optional) Specifies any customization for the newly cloned virtual machine. **Note:** In this release, the customization parameter must be of type autoprep. Otherwise, customization will fail. See Guest Customization Datatypes on page 61 for complete information on autoprep and the guest customization specification. Refer to the *Virtual Infrastructure SDK Programming Guide* for Java and Perl samples illustrating guest customization for Windows and Linux virtual machines.
autopoweron	xsd:boolean	(Optional) Flag that determines whether or not the newly cloned virtual machine will automatically power on once the cloning operation is complete. If the flag is set to true, then the virtual machine powers on automatically.

Output Message — CloneVMResponseMsg

Once the operation has been initiated, the request returns a task handle to the client. The client may monitor the task for the progress of the operation.

CloneVMResponse		
Name	**Type**	**Description**
returnval	ViewContents	The response specifies the ViewContents of the task performing the CloneVM operation: • ViewContents.handle — Handle for the task object. • ViewContents.vHandle — Versioned handle for the task object. • ViewContents.body — XML document of type Task that describes the task object.

Fault

FaultInfo		
Name	**Type**	**Description**
kind	FaultKind	• BadRequest — The request is not well formed. • HandleFault – The handle, passed as a parameter, has encountered errors. • NotFound – The handle that is passed cannot be found. • OperationFailed — The server could not successfully complete this operation.
code	xsd:int	Integer, that gives the error code that is similarly classified to HTTP error codes.
info	xsd:string	See the FaultInfo datatype for details.
val	xsd:anyType	See the FaultInfo datatype for details.

CreateTemplate

The CreateTemplate operation creates a new template from an existing virtual machine (source).

Permissions

The client must have Configure rights on the virtual machine used to create the template.

Input Message — CreateTemplateRequestMsg

CreateTemplate		
Name	**Type**	**Description**
srcHandle	xsd:string	Handle to the (source) virtual machine used to create the template.
info	TemplateSpec	Specifies the following set of information: • name — Name of the template that is created. • description — Description of the template, in your own words. • datastore — (Optional) Location where the template's virtual disk(s) and configuration file resides. The VMware VirtualCenter Server Repository datastore is the default location.

Output Message — CreateTemplateResponseMsg

Once the operation has been initiated, the request returns a task handle to the client. The client may monitor the task for the progress of the operation.

CreateTemplateResponse		
Name	**Type**	**Description**
returnval	ViewContents	The response specifies the ViewContents of the task performing the CreateTemplate operation: • `ViewContents.handle` — Handle for the task object. • `ViewContents.vHandle` — Versioned handle for the task object. • `ViewContents.body` — XML document of type Task that describes the task object.

Fault

FaultInfo		
Name	**Type**	**Description**
kind	FaultKind	• BadRequest — The request is not well formed. • HandleFault – The handle, passed as a parameter, has encountered errors. • NotFound — The input handle cannot be found. • OperationFailed — The server could not successfully complete this operation.
code	xsd:int	Integer, that gives the error code that is similarly classified to HTTP error codes.
info	xsd:string	See the FaultInfo datatype for details.
val	xsd:anyType	See the FaultInfo datatype for details.

RunTask

The RunTask operation starts an existing task schedule. This operation takes the handle of an existing task schedule that is to run as its input parameter,. An active task, corresponding to the task schedule, is immediately created under `/task`. Currently, clients cannot directly get the handle of the created task. Clients must search for the task containing the handle of the task schedule.

Permissions

The client must have Interact rights for the specified task.

Input Message — RunTaskRequestMsg

RunTask		
Name	**Type**	**Description**
task	xsd:string	Handle specifying the desired task.

Output Message — RunTaskResponseMsg

Upon success, an empty response message is returned.

Fault

FaultInfo		
Name	**Type**	**Description**
kind	FaultKind	• BadRequest — The request is not well-formed. • BadState — The task is already running or completed. • HandleFault – The handle, passed as a parameter, has encountered errors. • NotApplicable — The specified handle does not refer to a task. • OperationFailed — The server could not successfully complete this operation.
code	xsd:int	Integer, that gives the error code that is similarly classified to HTTP error codes.
info	xsd:string	See the FaultInfo datatype for details.
val	xsd:anyType	See the FaultInfo datatype for details.

EndTask

The EndTask operation stops a running task, or cancels a task that has not yet been started. The handle of the task that will be stopped is passed as an input parameter to this operation.

Note: Clients cannot cancel all tasks. Clients can only use the EndTask operation to cancel tasks that have the allowCancel boolean of the Task datatype set to true.

Permissions

The client must have Interact rights for the specified task.

Input Message — EndTaskRequestMsg

EndTask		
Name	**Type**	**Description**
task	xsd:string	Handle specifying the running task.

Output Message — EndTaskResponseMsg

Upon success, an empty response message is returned.

Fault

FaultInfo		
Name	**Type**	**Description**
kind	FaultKind	• BadRequest — The request is not well-formed. • BadState — The task is already running or completed. • HandleFault – The handle, passed as a parameter, has encountered errors. • NotApplicable — The specified handle does not refer to a task. • OperationFailed — The server could not successfully complete this operation.
code	xsd:int	Integer, that gives the error code that is similarly classified to HTTP error codes.
info	xsd:string	See the FaultInfo datatype for details.
val	xsd:anyType	See the FaultInfo datatype for details.

CHAPTER 7

Sample Code

All sample code that is provided as part of the VMware SDK kit is subject to the following copyright.

Copyright Information

Each sample program included with the VMware SDK includes a copyright. However, for brevity, we do not include this copyright in its entirety with each sample program. Instead, we include the first line of the copyright followed by ellipses, to indicate its placement. The complete copyright is as follows:


```
Copyright (c) 1998-2005 VMware, Inc.

Permission is hereby granted, free of charge, to any person obtaining a
copy of the software in this file (the "Software"), to deal in the
Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included
in all copies or substantial portions of the Software.

The names "VMware" and "VMware, Inc." must not be used to endorse or
promote products derived from the Software without the prior written
permission of VMware, Inc.

Products derived from the Software may not be called "VMware", nor may
"VMware" appear in their name, without the prior written permission of
VMware, Inc.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
VMWARE,INC. BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

CHAPTER 8

Glossary

Access Control List — An access control list is a set of <group, rights> pairs that defines the access rights for an object.

Container — A Container describes an object in a hierarchy, such as a Server Farm in VMware VirtualCenter.

Farm — A Farm is a set of host machines and VirtualMachineGroups that collectively act as a single host. This concept is similar to a "Server Farm" in VMware VirtualCenter.

File — A file contains raw data, such as an image, or other auxiliary information.

FQDN — Fully Qualified Domain Name of a host includes both the host name and the domain name. For example, a host named `esx1` in the domain `vmware.com` is represented as `esx1.vmware.com`.

Group — A group is a set of users and groups.

Handle — A handle is a temporary token, used by a Web service client, to invoke Web service operations that require a reference to an object. An object handle is somewhat analogous to a file handle (descriptor) returned by a operating system similar to UNIX, when a file is opened using the "open" system call. Like a file handle, an object handle is a temporary handle that always refers to the same object.

Host or Host Machine — A host machine is a system capable of managing and executing virtual machines.

ISV – Independent Software Vendor — Systems management vendors, enterprise management frameworks, imaging and provisioning vendors, storage management vendors, and so on.

Item — An Item is a <name, value> pair for an element of information about an object. An Item is normally represented as an element in XML. However, in some cases, if may be necessary to represent a collection of Items in a single element for performance reasons.

Management Interface — The VMware Management Interface (management interface) is a Web-based management tool that allows you to monitor the state of virtual machines and the ESX Server machine on which they are running. It also allows you to control (power on, suspend, resume, reset, and power off) the virtual machines on ESX Server.

Name — A name is either a path that refers to an object (like a URL or URI in Web terms) or the name of an Item of information in the Web service.

Object — An object has a type associated with it and is described by an XML document, that is the value of the object. If you use the file system analogy, then Container, Farm, and VirtualMachineGroup are comparable to directories in a file system while all the other objects are comparable to individual files in a file system.

Path — A path resembles a full path name of a file, which point to or identifies an object in the Web service hierarchy.

Remote console — An interface to a virtual machine that provides non-exclusive access to a virtual machine from both the host on which the virtual machine is running, and from workstations connected to that host.

Service console — The VMware Service Console provides ESX Server system management functions and interfaces. It is installed as the first component, and is used to bootstrap the ESX Server installation and configuration. It also boots the system and initiates execution of the virtualization layer and resource manager. In ESX Server, the service console is implemented by using a modified Linux® distribution.

Service Host — The Web service executes on the service host.

Session token — Upon successful login, the Web service issues a token and as part of the HTTP header, passes it to the client in the response. Specifically the token is a cookie that should be passed back by the client with each request. The token identifies the session to the Web service and its format is opaque to the client. Session tokens can be passed across multiple connections to the Web service. A session token expires after a period of inactivity, and may also expire after a certain period of time, even if it is being actively used.

Target — A target is the object that corresponds to a request URI.

Uptime — Uptime is the total elapsed time since the host or virtual machine was last restarted. Uptime may be computed using the last bootTime or by looking up the interval from the cumulative PerfSample.

UUID — Universally Unique Identifier (ID). This is a 128-bit number represented in hexadecimal (HEX) format when passed as a string; for example, `f81d4fae-7dec-11d0-a765-00a0c91e6bf6`.

User — A user is a principal known to the system.

vHandle — A vHandle as a reference to the specific memory state of an object at a certain point in time. That is, a vHandle is an object handle that has a version number associated with it. The version number determines the specific memory state.

VirtualCenter — VMware VirtualCenter is a software solution for deploying and managing virtual machines across the data center.

Virtual Machine — A virtual machine contains system and configuration state for a machine and may execute on a host machine.

Virtual Machine Array — A virtual machine array is a set of virtual machines that may be operated on collectively. This concept is currently called a "VM Group" or "VM Folder" in VMware VirtualCenter.

VMA — The VMA is the VMware VirtualCenter Web Service, which provides a Web services interface that client programs may use to talk, by using the SOAP protocol.

Web Service — A programming interface based on SOAP and WSDL.

APPENDIX A

Virtual Machine Configuration Variables

VirtualMachineInfo.guestInfo is the list of configuration variables that describe the virtual machine and its guest operating system. These configuration variables are listed in the following table.

Note: ESX Server does not support any guest operating systems for which defaultDiskControllerType is set to `ide`.

The list of guest operating systems supported by ESX Server and GSX Server is not the same. Refer to *www.vmware.com/support/guestnotes/doc* for a complete list of guest operating systems supported by a particular host. Use this information when selecting a guest operating system for a virtual machine.

Name	Value	Description
guestOS — This variable describes the type of guest operating system. This guest operating system type is used during a Create virtual machine operation to determine the defaults, including the virtual machine's memory size, and so on.	dos	MS-DOS®
	win31	Windows 3.1
	Win-95	Windows 95
	win98	Windows 98
	winMe	Windows ME
	winNT	Windows NT®
	win2000Pro	Windows 2000 Professional
	win2000Serv	Windows 2000 Server
	win2000AdvServ	Windows 2000 Advanced Server
	winXPHome	Windows XP Home Edition
	winXPPro	Windows XP Professional
	winNetWeb	Windows Server™ 2003, Web Edition
	winNetStandard	Windows Server 2003, Standard Edition
	winNetEnterprise	Windows Server 2003, Enterprise Edition
	linux	Linux®
	freeBSD	FreeBSD
	netware5	Novell® NetWare® 5
	netware6	NetWare 6
	other	Other

APPENDIX B

VMware VirtualCenter Events

This appendix includes a complete list of all VirtualCenter 1.1 event messages. The event message format includes variables, enclosed by angled brackets; for example, `<hostname>`. The text in the angled brackets is replaced by the actual value at run time. For example, the message `<virtual machine name> on host <host name> is powered off` changes to something similar to `DXUU-WIN2000 on host DXUUU-ESX is powered off`.

For some VirtualCenter event messages, you see `<message>` in the format. This means that a message is being passed from an ESX Server host. We do not have a list of these messages at this time.

Each event message is associated with a type: error, warning, info, or user. Some of the event messages can be of multiple types: error, warning, or info. However, most event messages are of one particular type, as shown in the following tables.

VirtualCenter Event Messages

The list of VirtualCenter event messages may be sorted into the following general categories. The individual messages are listed in alphabetical order.

ESX Server Messages About the Host or Virtual Machine

Type	Message
info/warning/error	Host <host name>: <message>
info/warning/error	VM <virtual machine name>: <message>
info/warning/error	Virtual Machine <virtual machine name>: <message>

Messages About the User

Type	Message
error	Failed to login user <user name>. Reason: Bad username/password
error	Failed to login user <user name>. Reason: Already authenticated
error	Failed to login user <user name>. Reason: No access
user	User log entry: <message>
info	User <user name> logged in
info	User <user name> logged out
info	VMware VirtualCenter <version> started

Host Connection Messages

Type	Message
info	Connected to <host name>
error	Connection failed for <host name>: Account has insufficient privileges
error	Connection failed for <host name>: Already being managed by <server name>
error	Connection failed for <host name>: Bad username/password
error	Connection failed for <host name>: Could not connect to host
error	Connection failed for <host name>: Could not resolve hostname
error	Connection failed for <host name>: Failed to configure management account
error	Connection failed for <host name>: Failed to install/upgrade vmware-ccagent

Type	Message
error	Connection failed for <host name>: Incompatible version
error	Connection failed for <host name>: Network error
error	Connection failed for <host name>: Not enough CPU licenses
error	Connection failed for <host name>: Server agent is not responding
error	Connection failed for <host name>: Timeout waiting for host response
error	Connection failed for <host name>: Unexpected error connecting to host
info	Disconnected from <host name>
error	Lost connection to <host name>
error	Reconnection failed for <host name>
info	Removed host <host name>
info	Shutdown of host <host name>, reason: <message>

Virtual Machine Messages

Type	Message
error	A MAC address (<address>) of <virtual machine name> conflicts with Virtual Machine <virtual machine name>
error	A static configured MAC address (<address>) of <virtual machine name> conflicts with Virtual Machine <virtual machine name>
info	Powered-on Virtual Machine <virtual machine name> being migrated from host <host name> to host <new host>
info	Removed <virtual machine name> from <host name>
info	<virtual machine name> discovered on host <host name>
warning	<virtual machine name> does not exist on host <host name>
info	<virtual machine name> on host <host name> is disconnected
info	<virtual machine name> on host <host name> is powered off
info	<virtual machine name> on host <host name> is powered on
info	<virtual machine name> on host <host name> is resetting
info	<virtual machine name> on host <host name> is resuming
info	<virtual machine name> on host <host name> is starting
info	<virtual machine name> on host <host name> is stopping
info	<virtual machine name> on host <host name> is suspended
info	<virtual machine name> on host <host name> is suspending
info	Virtual Machine <virtual machine name> being created on host <host name>

Type	Message
info	Virtual Machine <virtual machine name> on <host name> has invalid name. Renamed from <old name> to <new name>

Cloning and Migration Messages

Type	Message
error	BIOS ID (<id>) of <virtual machine name> conflicts with that of <virtual machine name>
info	Changed resource allocation for Virtual Machine <virtual machine name>
error	Date of this machine has been rolled back. Disconnecting all hosts.
error	Failed to clone Virtual Machine <virtual machine name> onto host <host name>. Reason: <message>
error	Failed to create virtual disk <disk name> for virtual machine <virtual machine name>: <message>
error	Failed to deploy template <template name> on host <host name>. Reason: <message>
error	Failed to migrate Virtual Machine <virtual machine name> from host <host name> to host <host name>. Reason: <message>
error	Failed to <power operation> for virtual machine <virtual machine name> on host <host name>: <message>
error	License of <name> has expired. Disconnecting all hosts. (Disabling VMotion as well.)
info	Migration of Virtual Machine <virtual machine name> from host <host name> to host <host name> completed
warning	Name for Virtual Machine <virtual machine name> updated on host. Renamed from <old name> to <new name>
error	Number of connected hosts has exceeded available host licenses. Disconnecting all hosts.
error	Number of VMotion-enabled hosts exceeds number of available licenses. Disabling VMotion on all hosts.
info	Powered-off Virtual Machine <virtual machine name> migrating from host <host name> to host <host name>
info	Powered-on Virtual Machine <virtual machine name> is migrating off host <host name>
info	Template <template name> being deployed on host <host name>
info	Template <template name> deployed on host <host name>

Type	Message
info	The Adapter <adapter> for Virtual Machine <virtual machine name> was assigned a new MAC address: <address>
warning	The destination host <host name> does not have access to the same networks as Virtual Machine <virtual machine name>.
warning	The MAC address for adapter <adapter> in Virtual Machine <virtual machine name> was changed from <old> to <new>
info	<virtual machine name> on host <host name> cloned from <virtual machine name> on host <host name>
info	<virtual machine name> on host <host name> is being cloned to <virtual machine name> on host <host name>

Task Messages

Type	Message
info	Task <scheduled task name> created on <entity name>
info	Task <scheduled task name> on <entity name> completed successfully
error	Task <scheduled task name> on <entity name> failed. Reason: <message>
error	Task <scheduled task name> on <entity name> failed to send notification to <server>. Reason: <message>
info	Task <scheduled task name> on <entity name> reconfigured
info	Task <scheduled task name> on <entity name> sent notification to <server>
info	Task <scheduled task name> on <entity name> started
info	Task <scheduled task name> removed from <entity name>

Alarm Messages

Type	Message
info	Alarm <alarm name> created on <entity name>
error	Alarm <alarm name> on <entity name> failed to run script <script>. Reason: <message>
error	Alarm <alarm name> on <entity name> failed to send e-mail to <email addresses>. Reason: <message>
error	Alarm <alarm name> on <entity name> failed to send snmp trap to <server>. Reason: <message>
info	Alarm <alarm name> on <entity name> ran script <script>

Type	Message
info	Alarm <alarm name> on <entity name> reconfigured
info	Alarm <alarm name> on <entity name> sent e-mail to <email addresses>
info	Alarm <alarm name> on <entity name> sent snmp trap to <server>
info	Alarm <alarm name> removed from <entity name>
info	<entity name> caused alarm <alarm name> on <entity name> to change from <color> to <color>
info	<entity name> caused action <entity name> of alarm <alarm name> on <entity name> to trigger

Storage Message

Type	Message
error	Multiple datastores named <data store name> detected on host <host name>

APPENDIX C

Performance Statistics

This appendix describes the SDK performance statistics, and how you may reconcile them with the performance charts in the VMware VirtualCenter application.

Calculating Performance Statistics

We discuss CPU, memory, and network performance statistics in this appendix.

CPU Statistics on the Host Summary Page

In this section, we describe how the SDK CPU performance statistics correlate with the VirtualCenter CPU performance chart for a host. There are 3 bar graphs:

- System Total — Total CPU performance for the host.
- Virtual machines — Total CPU performance for all running virtual machines.
- Other — Difference of Virtual machines subtracted from System Total.

Note: CPU performance statistics are in milliseconds. The calculations in this section refer to both the CPUPerf and CPUPerf2 datatypes, and are rounded off to the nearest integer.

System Total Bar Graph

The host system CPU total is calculated by the following equation:

System total = Sum(CPUPerf.used) / (1000 * interval * nHostPhysicalCPU)

where Sum(CPUPerf.used) is the sum of CPUPerf.used for each physical CPU on the host and nHostPhysicalCPU is the number of physical CPUs on the host.

For example, if the host has 2 physical CPUs:

System total = (CPUPerf.used [PCPU1] + CPUPerf.used [PCPU2]) / (1000 * interval * 2)

where PCPU is the physical CPU on the host.

Virtual Machines Bar Graph

Virtual Machines represents the total for all the running virtual machines on the host:

Virtual Machines = Virtual Machine[1] + Virtual Machine[2] + ... Virtual Machine[n]

The value for each Virtual Machine[n] is calculated by the following equation:

Virtual Machine[n] = Sum(CPUPerf.used) / (1000 * interval * nHostPhysicalCPU)

where Sum(CPUPerf.used) is the sum of CPUPerf.used for each virtual CPU on the virtual machine and nHostPhysicalCPU is the number of physical CPUs on the host.

For example, if a virtual machine has two virtual CPUs, on a host with 2 physical CPUs:

Virtual Machine[1] = (CPUPerf.used [VCPU1] + CPUPerf.used [VCPU2]) / (1000 * interval * 2)

where VCPU is the virtual CPU on the virtual machine.

Similarly, if a second virtual machine has one virtual CPU:

Virtual Machine[2] = CPUPerf.used [VCPU1] / (1000 * interval * 2)

Other Bar Graph

Other represents the difference when subtracting Virtual Machines from Host system total:

Other = System Total - Virtual Machines

Memory Performance Statistics

The MemoryPerf, MemoryPerf2, VirtualMachineMemoryPerf, and VirtualMachineMemoryPerf2 datatypes are actual numbers, in KB. No calcuations are applied to these memory performance statistics.

Disk, Generic Device, and Network Performance Statistics

The DiskPerf, GenericDevicePerf, and NetworkPerf datatypes are all calculated in a similar manner.

To obtain a result in KB/second, use the following equation:

KB/sec = Perf.bytes/ (1024 * PerfStats.source.sample.interval)

where Perf.bytes is DiskPerf.bytesRd or DiskPerf.bytesWr, or GenericDevicePerf.bytesRd or GenericDevicePerf.bytesWr, or NetworkPerf.bytesIn or NetworkPerf.bytesOut.

To obtain a result in blocks/second for DiskPerf, use the following equation:

blocks/sec = DiskPerf.num/ PerfStats.source.sample.interval

where DiskPerf.num is either DiskPerf.numRd or DiskPerf.numWr.

Similarly, to obtain a result in packets/second for NetworkPerf, use the following equation:

packets/sec = NetworkPerf.num/ PerfStats.source.sample.interval

where NetworkPerf.num is either NetworkPerf.packetsIn or NetworkPerf.packetsOut.

APPENDIX D

Valid TimeZone Indexes

The following table lists valid TimeZone indexes values for use by Sysprep. You may also obtain this list from Microsoft. As of this writing, the valid URL for the TimeZone table is *www.microsoft.com/technet/prodtechnol/windows2000pro/deploy/unattend/sp1ch01.mspx*.

Index	TimeZone
000	Int'l Dateline
001	Samoa
002	Hawaii
003	Alaskan
004	Pacific
010	Mountain (U.S. and Canada)
015	U.S. Mountain: Arizona
020	Central (U.S. and Canada)
025	Canada Central
030	Mexico

Index	TimeZone
033	Central America
035	Eastern (U.S. and Canada)
040	U.S. Eastern: Indiana (East)
045	S.A. Pacific
050	Atlantic (Canada)
055	S.A. Western
056	Pacific S.A.
060	Newfoundland
065	E. South America
070	S.A. Eastern
073	Greenland
075	Mid-Atlantic
080	Azores
083	Cape Verde Islands
085	GMT (Greenwich Mean Time)
090	GMT Greenwich
095	Central Europe
100	Central European
105	Romance
110	W. Europe
113	W. Central Africa
115	E. Europe
120	Egypt
125	EET (Helsinki, Riga, Tallinn)
130	EET (Athens, Istanbul, Minsk)
135	Israel: Jerusalem
140	S. Africa: Harare, Pretoria
145	Russian
150	Arab
155	E. Africa
160	Iran
165	Arabian
170	Caucasus Pacific (U.S. and Canada)

Index	TimeZone
175	Afghanistan
180	Russia Yekaterinburg
185	W. Asia
190	India
193	Nepal
195	Central Asia
200	Sri Lanka
201	N. Central Asia
203	Myanmar: Rangoon
205	S.E. Asia
207	N. Asia
210	China
215	Singapore
220	Taipei
225	W. Australia
227	N. Asia East
230	Korea: Seoul
235	Tokyo
240	Sakha Yakutsk
245	A.U.S. Central: Darwin
250	Central Australia
255	A.U.S. Eastern
260	E. Australia
265	Tasmania
270	Vladivostok
275	W. Pacific
280	Central Pacific
285	Fiji
290	New Zealand
300	Tonga

APPENDIX E

Revision History

The following table lists the revision history for the *Virtual Infrastructure SDK Reference Guide*.

Date	Description
October 25, 2005	Corrected the description of the RevertVM operation to state that this operation is only supported on GSX Server.
September 29, 2005	Corrected the description of the soft flag for the StopHost operation. Updated samples copyright to 2005.
September 22, 2005	Clients can now enable or disable VMotion migrations on a host. Updated HostInfo and MigrationInfo datatypes. Selected fields are now modifiable by the PutUpdates operation.
November 30, 2004	GA version of this guide.

Index

A

Access control list **23**, **197**

Access rights **76**

Adapter **69**

Administer right **168**

Advantage of Web services **16**

Affinity **77**
- CPU **39**
- node **77**

Alert **54**

AlreadyExists fault **58**, **59**, **153**, **159**

Answer question from virtual machine **133**–**134**

AnswerVMRequestMsg **133**

AnswerVMResponseMsg **133**

Architecture
- NUMA **21**, **72**–**75**, **77**

Array **41**

Array of Items **47**

Automatic tasks **32**

Automatically powering on a virtual machine after cloning **188**

B

BadHandle fault **71**

BadName fault **58**, **59**

BadRequest fault **58**, **59**

BadState fault **58**, **59**

BadVersion fault **71**

Blocked virtual machine **133**–**134**

Blocking GetUpdates **141**

Blocking virtual machine **112**

Boot time for a virtual machine **104**

Boot time of host machine **75**

bootTime value **25**

Browse right **118**, **120**, **123**, **144**, **158**, **164**, **166**

C

CancelGetUpdates **141**

CancelGetUpdatesRequestMsg **144**

Cancelling a task **193**

CD-ROM drive **72**

Change
- edit **44**–**45**
- element **142**
- object **41**–**43**
- operation **43**–**46**, **146**
- requisition **46**

ChangeConflict fault **58**, **59**, **147**

ChangeOp **42**

ChangePermissionsRequestMsg **168**

ChangeReq **41**, **146**

ChangeReqList **41**

Changes for a vHandle **23**

Changing data **146**–**147**

Changing the resource settings of a virtual machine **150**

Client tasks **22**

CloneVM **150**

CloneVMRequestMsg **187**–**188**

CloneVMResponseMsg **188**

Collecting event information **54**–**57**

Collection of data samples **164**, **166**

Committing changes to virtual disk **172**–**173**

Composite target for Change datatype **41**

Configuration parameters **27**

Configuration variables for a virtual machine **202**

Configure right **139**, **146**, **148**, **156**, **158**, **162**, **170**, **172**, **174**, **179**, **181**, **182**, **185**, **187**, **190**

Configuring
- CPU **28**
- disk resources **28**
- guest operating system **187**–**189**
- memory **28**, **78**–**79**
- networking **28**
- server **21**
- virtual machine **21**, **27**, **127**

Connecting to a host machine **74**

ConsolidateVMRequestMsg **172**

ConsolidateVMResponseMsg **172**
Container **23**, **47**, **57**, **197**
  changing permissions **168**–**169**
  creating **148**–**155**
  deleting **162**–**163**
  GetContents operation on **121**
  renaming **158**–**161**
Controller **48**, **95**
Core data model **21**, **23**
CPU **25**, **39**–**41**, **72**
  affinity set **39**
  performance **40**
  performance for a virtual machine **98**
  shares **39**
  time **40**
  utilization **40**
  virtual **39**, **93**, **98**
  virtual machine **28**
CPU utilization **128**
CPUControls **28**
CPUInfo **25**
Create **90**, **152**, **179**, **181**
CreateRequestMsg **149**–**150**
CreateResponseMsg **153**
CreateTemplate **151**
CreateTemplateRequestMsg **190**
CreateTemplateResponseMsg **190**
CreateVirtualDiskRequestMsg **156**
CreateVirtualDiskResponseMsg **157**
Creating
  task **150**
  virtual machine from existing one **187**–**189**
  virtual machine from template **22**, **33**, **187**–**189**
Creating virtual disk **156**–**157**
Custom property **48**
Customizing a virtual machine **187**–**189**
Customizing guest operating system **22**

D

Data, changing **146**–**147**
dataLocator **182**–**183**
Declarations, event **31**
Delete change operation **45**, **142**
DeleteRequestMsg **162**
destHandle **159**
Devices **60**
  generic **28**, **60**–**61**
  network interface name **81**
Disabled state **151**, **179**
DisableHostRequestMsg **181**
DiscoveredVMs **179**
Disk **72**
  moving virtual **182**–**184**
  performance data **53**
  resource management **50**
  shares **50**
  virtual **52**–**54**, **94**–**97**, **170**–**171**
  virtual disk structure **54**
Disk utilization **128**
diskInfo parameter **156**
Domain name **73**
DuplicateHandle fault **71**

E

Edit change operation **44**–**45**, **142**
Element management operations **17**–**18**
EnableHost **151**
EnableHostRequestMsg **179**
Error **57**–**60**
Event **31**, **54**–**57**, **128**
  arguments **31**
  logs **21**
  on host machine **72**–**75**
Event collector **106**
Event declarations **31**
Event Log data model **21**, **31**
EventArgInfo **31**
EventCollection **31**
EventDecl **31**
EventDeclArray **31**
EventLog **31**

F

Farm **57**, **197**
  creating **148**–**155**

deleting **162**–**163**
GetContents operation on **121**
renaming **158**–**161**
Fault **57**–**60**
CancelGetUpdates **144**–**145**
ChangePermissions **168**
CloneVM **189**
Create **153**
CreateTemplate **191**
CreateVirtualDisk **157**
Delete **163**
DisableHost **181**
EnableHost **180**
EndTask **193**
GetContents **122**
GetInfo **123**
GetUpdates **143**
Login **116**
MigrateVM **186**
MoveVM **184**
PutUpdates **147**
QueryPerfData **165**, **167**
Rename **159**
ResetVM **138**
ResolvePath **118**
RunTask **192**
StartVM **132**
StopHost **139**
StopVM **136**
File **197**
Filter for performance collector **83**
Floppy drive **72**
FQDN **73**, **197**
Fully Qualified Domain Name.See FQDN.
G
Generic devices **60**–**61**
GenericDeviceInfo **28**
GetContents **23**, **48**, **71**, **133**
GetContentsRequestMsg **120**
GetContentsResponseMsg **120**
GetInfo **93**
GetInfoRequestMsg **123**
GetInfoResponseMsg **123**
GetUpdates **41**, **131**, **133**, **135**, **137**, **168**
cancelling operation **144**–**145**
GetUpdatesRequestMsg **141**
GetUpdatesResponseMsg **142**
Group **197**
Guest customization data model **22**
Guest operating system **68**, **69**–**70**, **99**, **104**, **107**–**108**, **202**
configuring **187**–**189**
customizing **22**
standby mode **135**
Guest operating system memory **78**
guestInfo parameter **27**
H
Handle **41**, **71**–**72**, **135**–**136**, **137**–**138**, **141**–**142**, **146**–**147**, **185**–**186**, **187**–**188**, **197**
HandleFault **58**, **59**
Hardware **25**, **72**, **72**–**75**
virtual **26**–**28**, **33**, **99**
Hardware, resetting virtual **137**–**138**
Heap memory **79**
Hierarchy of Web service **21**
Host **198**
adding **151**
changing permissions **168**–**169**
creating **148**–**155**
deleting **162**–**163**
Disabled state **179**
disabling **181**
enabling **179**–**180**
restart **139**–**140**
shut down **139**–**140**
Host machine **25**, **72**–**75**
boot time **75**
configuration **21**
connecting to **74**
CPU **39**–**40**
event **54**
event logs **21**
hardware **72**
memory **25**, **78**–**79**
name **73**
network interface **25**
networking **81**
operating system **25**
operating system version **73**

performance **84**
performance metrics **29**
peripherals **25**, **72**
resource utilization **29**
status **75**
Host Machine data model **21**, **24**–**25**
HostInfo **25**
HostState **25**
Hyper-threading **39**

I

IDE controller **48**
Indexed array **42**
Input message
AnswerVMRequestMsg **133**
CancelGetUpdatesRequestMsg **144**
ChangePermissionsRequestMsg **168**
CloneVMRequestMsg **187**–**188**
CreateRequestMsg **149**–**150**
CreateTemplateRequestMsg **190**
CreateVirtualDiskRequestMsg **156**
DeleteRequestMsg **162**
DisableHostRequestMsg **181**
EnableHostRequestMsg **179**
EndTaskRequestMsg **193**
GetContentsRequestMsg **120**
GetInfoRequestMsg **123**
GetUpdatesRequestMsg **141**
LoginRequestMsg **115**
MigrateVMRequestMsg **185**
MoveVMRequestMsg **183**
PutUpdatesRequestMsg **146**
QueryPerfDataRequestMsg **164**, **166**
RenameRequestMsg **158**
ResetVMRequestMsg **137**
ResolvePathRequestMsg **118**
RunTaskRequestMsg **192**
StartVMRequestMsg **131**
StopHostRequestMsg **139**
StopVMRequestMsg **135**
Insert change operation **45**, **142**
Interact right **131**, **133**, **135**, **137**, **141**, **192**
Interrupted fault **60**, **143**, **144**
Interval, sample **83**–**84**
ISV **198**
Item **76**, **198**
Item array **47**

K

Kernel information for a host machine **72**–**75**
Key-based array **42**

L

Leaf value target for Change datatype **41**
Leaf value, edit change operation **44**–**45**
List of events **31**
Log of events **31**
Logging for virtual machine **100**
Logical name
of device **60**–**61**
of network interface **81**
LoginRequestMsg **115**
LoginResponseMsg **115**

M

Maximum memory for a virtual machine **77**
Maximum processor percentage for a virtual machine **39**
Memory **25**, **72**, **78**–**79**, **170**–**171**
available for creating virtual machine **47**, **92**
performance **78**
performance for a virtual machine **103**
shares **77**
utilization **128**
virtual machine **28**, **77**, **108**
MemoryControls **28**
MemoryInfo **25**
Message from virtual machine **112**
Message, event **31**
MigrateVM **150**, **159**
MigrateVMRequestMsg **185**
MigrateVMResponseMsg **185**
MigrationInfo **25**
Minimum memory for a virtual machine **77**
Minimum processor percentage for a

virtual machine **39**
Model
  Core data **21**, **23**
  Event Log data **21**, **31**
  Guest customization specification **22**
  Host Machine data **21**, **24**–**25**
  Performance Metric data **21**, **29**–**30**
  Task data **22**, **32**
  Template data **22**, **33**
  Virtual Machine data **21**, **26**–**28**
Move change operation **45**, **142**
MoveVM **150**
MoveVMRequestMsg **183**
MoveVMResponseMsg **183**
Moving an object in the /vcenter hierarchy **158**–**161**
Moving virtual disks **182**–**184**

N

Name **198**
Network adapter **69**
Network interface card. See NIC.
Network traffic shaping **81**
Network utilization **128**
Network, virtual **108**–**110**
NetworkControls **28**
NetworkInfo **25**
Networking **25**, **72**, **81**–**82**
  in virtual machine **28**
  performance **81**
nextVHandle **144**
nfshaper **81**
NIC **28**
Node affinity set **77**
Non-string leaf value **44**–**45**
Non-uniform memory architecture. See NUMA.
NoResources fault **60**
NotApplicable fault **58**, **60**
NotDirectory fault **58**, **60**
NotFound fault **58**, **60**
NUMA architecture **21**, **72**–**75**, **77**

O

Object **198**
  container **23**, **47**, **57**
ObsoleteHandle fault **72**
Operating system on host machine **25**, **73**
OperationFailed fault **58**, **60**
Output Message
  AnswerVMResponseMsg **133**
Output message
  CloneVMResponseMsg **188**
  CreateResponseMsg **153**
  CreateTemplateResponseMsg **190**
  CreateVirtualDiskResponseMsg **157**
  EndTaskResponseMsg **193**
  GetContentsResponseMsg **120**
  GetInfoResponseMsg **123**
  GetUpdatesResponseMsg **142**
  LoginResponseMsg **115**
  MigrateVMResponseMsg **185**
  MoveVMResponseMsg **183**
  PutUpdatesResponseMsg **147**
  QueryPerfDataResponseMsg **164**, **166**
  ResetVMResponseMsg **137**
  ResolvePathResponseMsg **118**
  StartVMResponseMsg **131**
  StopVMResponseMsg **135**

P

Parameter
  config **27**
  guestInfo **27**
  task **32**
Parent performance collector **29**, **83**
Password for authenticating Web service to host machine **74**
Path **76**, **198**
PerfCollection **29**
PerfCollector. See Performance collector.
PerfFilter **29**
Performance **81**–**86**
  data for a generic device **60**–**61**
  data for virtual disk **53**
  metrics **29**
  on the host machine **72**–**75**

sample for data collection **83**–**84**
statistics **82**
virtual machine **128**
Performance collector **29**, **82**–**84**
creating **148**–**155**
filter **83**
requesting data from **164**–**167**
Performance Metric data model **21**, **29**–**30**
PerfSample **29**
PerfSource **29**
PerfSourceType **30**
PerfStat **30**
PerfType **30**
Peripherals **25**, **72**
Permission **76**
AnswerVM **133**
CancelGetUpdates **144**
ChangePermissions **168**–**169**
CloneVM **187**
ConsolidateVM **172**
Create **148**
CreateTemplate **190**
CreateVirtualDisk **156**
delete **162**
DisableHost **181**
EnableHost **179**
EndTask **193**
GetContents **120**
GetInfo **123**
GetUpdates **141**
Login **115**
MigrateVM **185**
MoveVM **182**
PutUpdates **146**
QueryPerfData **164**
QueryPerfData2 **166**
Rename **158**
ResetVM **137**
ResolvePath **118**
RevertVM **174**
RunTask **192**
SnapshotVM **170**
StartVM **131**
StopHost **139**
StopVM **135**
PermissionDenied fault **58**, **60**
Permissions
changing **168**–**169**
Picture **170**–**171**
Power operations
Creating a task for **150**
virtual machines **127**
Powering on virtual machine automatically after cloning **188**
Pressure, memory **79**
Processor
affinity set **39**
CPU **39**–**40**
host **72**
information **25**
time **40**
virtual **28**
Property, custom **48**
PutUpdates **41**, **59**, **101**, **141**, **150**
PutUpdatesRequestMsg **146**
PutUpdatesResponseMsg **147**

Q

QueryPerfDataRequestMsg **164**, **166**
QueryPerfDataResponseMsg **164**, **166**

R

Redo log **172**–**173**
Redo logs **170**
RemovableDeviceInfo **28**
RenameRequestMsg **158**
Replace change operation **45**, **142**
Request, error during **57**–**60**
Requesting data from a performance collector **164**–**167**
Resetting a virtual machine **137**–**138**
ResetVMRequestMsg **137**
ResetVMResponseMsg **137**
ResolvePath **60**, **76**
ResolvePathRequestMsg **118**
ResolvePathResponseMsg **118**
Resource management
CPU **28**
disk **28**, **50**
memory **28**
networking **28**
virtual CPU **28**

virtual machine memory **77**
Resource utilization **29**
Restarting the host **139**–**140**
Resuming a virtual machine **151**
Reverting virtual machine to a previous state **174**–**175**
RevertVMRequestMsg **174**
RevertVMResponseMsg **174**
RunTaskRequestMsg **192**

S

Sample collection **164**, **166**
Sample for performance data collection **29**, **83**–**84**
Schedule, task **90**–**91**
Scheduling tasks **32**
Script for virtual machine **107**
Scripts for virtual machine state changes **27**
SCSI controller **48**
Server **25**, **72**–**75**
  boot time **75**
  configuration **21**
  connecting to **74**
  CPU **39**–**40**
  event **54**
  event logs **21**
  hardware **72**
  memory **78**–**79**
  name **73**
  networking **81**
  operating system **25**
  operating system version **73**
  performance **84**
  performance metrics **29**
  peripherals **25**, **72**
  status **75**
Server Farms Groups **47**
Service console **72**–**75**
Session token **198**
Shared guest operating system memory **78**
Shares
  CPU **39**
  disk **50**
  memory **77**
Shutting down the host **139**–**140**
SMBIOS UUID **73**, **99**
Snapshot **170**–**171**, **174**–**175**
SnapshotVMRequestMsg **170**
SnapshotVMResponseMsg **171**
Source of performance statistics **29**
Standby mode for a guest operating system **135**
Start time for sample collection **164**, **166**
Starting task schedule **192**
StartVM **151**
StartVMRequestMsg **131**
StartVMResponseMsg **131**
State
  host **25**
  virtual machine **27**
Statistics
  performance **29**, **82**
  Web service **87**
Status
  host **75**
  host machine **72**–**75**
  memory **78**–**79**
  task **32**
StopHostRequestMsg **139**
Stopping a running task **193**
Stopping a virtual machine **135**–**136**
StopVM **151**
StopVMRequestMsg **135**
StopVMResponseMsg **135**
String leaf value **44**
Stub class
  TaskScheduleSpec **151**
Suspend directory **100**
Suspending a virtual machine **135**–**136**
Swap space **78**
System alert **54**
System heap memory **79**
System management BIOS UUID **73**
System time **40**
SystemFailure **114**
SystemInfo **25**

T

Target **199**

Task **32**, **88**–**91**, **135**–**136**, **137**
  cancelling **193**
  creating **150**
  status **32**
  stopping **193**
  weekly **111**

Task data model **22**, **32**

Task schedule
  creating **148**–**155**
  starting **192**

Tasks by client **22**

TaskSchedule **32**

TaskScheduleSpec **151**

Template data model **22**, **33**

Template, creating **190**–**191**

Template, creating virtual machine from **187**–**189**

Templates **22**, **33**

Time
  CPU **40**
  system **40**

Token, session **198**

Tools, VMware **27**

Traffic shaping, network **81**

Type of device **60**–**61**

U

Unauthorized fault **72**

Universally Unique Identifier (ID). See UUID.

UnknownRequest **114**

unknownVM directory **179**

Unreserved
  swap space **78**

Unreserved memory **78**

Update **23**

Updating data **146**–**147**

Uptime **199**

User **199**

User-defined property **48**

Username for authenticating Web service to host machine **74**

Utilization of
  CPU **40**
  resources **29**

UUID **73**, **127**, **179**, **199**

V

Versioned handle. See vHandle.

vHandle **71**–**72**, **92**, **120**, **141**–**142**, **144**, **146**–**147**, **168**, **184**, **186**, **199**
  changed object **41**
  list of changes **23**

ViewContents **23**

Virtual computing operations **18**

Virtual CPU **28**, **39**, **93**, **98**
  utilization **40**

Virtual disk **52**–**54**, **170**–**171**
  committing changes **172**–**173**
  creation **156**–**157**
  information **28**
  moving **182**–**184**
  performance data **53**
  resource management **50**
  structure **54**

Virtual hardware **26**–**28**, **33**, **99**
  resetting **137**–**138**

Virtual machine **27**, **98**–**108**, **156**–**157**, **199**
  answering question **133**–**134**
  blocking question **112**
  changing permissions **168**–**169**
  changing resource settings **150**
  configuration variables **202**
  configuring **21**, **127**
  CPU **28**
  CPU performance **98**
  creating **148**–**155**
  creating by using existing one **187**–**189**
  creating from template **22**, **33**, **187**–**189**
  customizing **187**–**189**
  deleting **162**–**163**
  discovering **179**–**180**
  event **54**
  event logs **21**
  maximum processor percentage **39**
  memory **28**, **77**, **108**, **170**–**171**
  memory performance **103**

migrating **25**, **185**–**186**
minimum processor percentage **39**
moving virtual disks **182**–**184**
moving when powered off **182**–**184**
networking **28**
NUMA architecture **21**
performance **84**, **128**
performance metrics **29**
power operations **127**
powering on automatically after cloning **188**
renaming **158**–**161**
resetting **137**–**138**
resource utilization **29**
resuming **131**–**132**, **151**
reverting to a previous state **174**–**175**
script **107**
scripts for state changes **27**
snapshot **170**–**171**
starting **131**–**132**
state **27**
stopping **135**–**136**
suspending **135**–**136**
template **22**, **33**
virtual disk **28**

Virtual Machine Array **199**

Virtual Machine data model **21**, **26**–**28**

Virtual machine group
creating **148**–**155**
deleting **162**–**163**
DiscoveredVMs **179**
GetContents operation on **121**
renaming **158**–**161**

Virtual machines on host machine **72**–**75**

Virtual NIC **28**

VirtualCenter **199**

VirtualCPUInfo **28**

VirtualDiskDestination **182**–**183**

VirtualDiskInfo **28**

VirtualMachineInfo **27**

VirtualMachineState **27**

VirtualMachineTools **27**

VirtualMemoryInfo **28**

VirtualNetworkAdapter **28**

VirtualNetworkInfo **28**

VMA **199**

VMotion **178**

VMware Tools **27**, **69**, **101**, **107**–**108**

W

Wait time **40**

Web service **16**, **199**
connecting to host machine **74**
detecting error during request **57**–**60**
hierarchy **21**
statistics **87**

X

XML document **198**