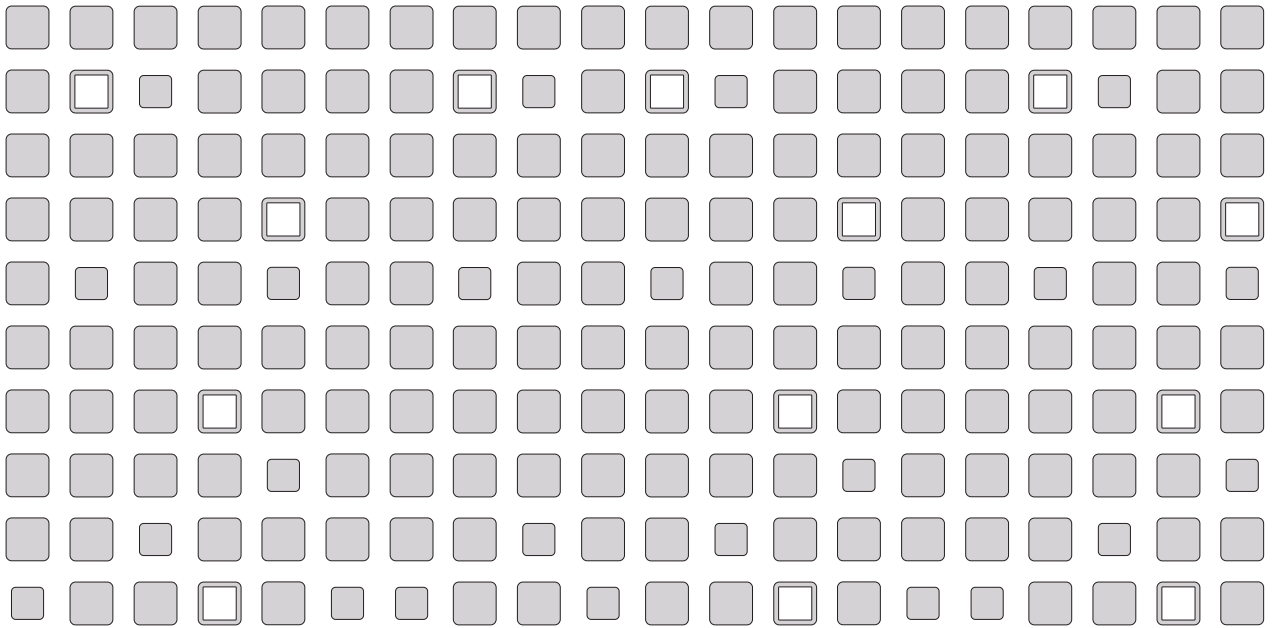


VERSION 2.0.1

VMware Infrastructure SDK

Porting Guide



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 © 1998-2006 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, 6,795,966, 6,880,022, 6,961,941, 6,961,806, and 6,944,699; patents pending. VMware, the VMware "boxes" logo and design, Virtual SMP and VMotion are registered trademarks or trademarks of VMware, Inc. in the United States and/or other jurisdictions. All other marks and names mentioned herein may be trademarks of their respective companies. Revision 20060922 Version 2.0.1 Item: SDK-ENG-Q306-283

Table of Contents

An Overview to Porting	5
Overview of the Porting Guide	6
Using This Porting Guide	8
Intended Audience	8
Managed Objects and Data Objects	9
Managed Object References: Accessing Managed Objects	9
Where to Find SDK 1.x Equivalents	11
Before Porting Your SDK 1.x Client Applications	12
Permissions	12
Logging On	13
Technical Support Resources	15
Standards and Reference Documentation	15
Handling Inventory	17
Getting Inventory Contents	18
Getting Updates on Inventory Contents	26
Creating Inventory	28
Moving Objects in Inventory	33
Renaming Objects in Inventory	34
Deleting Inventory	35
Virtual Machines and Hosts	37
Creating a Virtual Machine	38
Creating a Virtual Machine from Scratch	38
Cloning and Deploying Virtual Machines	40
Creating a Template	41
Creating a Virtual Machine from a Template	42
Creating a Virtual Disk	43
Provisioning a Virtual Machine	44
Updating Configurations	46
Connecting and Disconnecting Hosts	48
Connecting the Host to VirtualCenter	48
Disconnecting the Host from VirtualCenter	49
Migrating Virtual Machines	50
Hot Migration	50
Cold Migration	51

Validating Migration _____	52
Moving Files _____	52
Deleting Hosts and Virtual Machines _____	54
Power Operations _____	55
Powering On a Virtual Machine _____	56
Powering Off a Virtual Machine _____	57
Suspending a Virtual Machine _____	59
Resetting a Virtual Machine _____	60
Event Messages _____	61
Mapping Events in SDK 1.x and SDK 2.0 _____	62
Corresponding Events Between 1.x and 2.0 _____	62
More Virtual MachineEvents _____	66
More Host Events _____	67
Formatting Event Messages _____	68
Performance Monitoring _____	71
Performance Monitoring in SDK 1.x and 2.0 _____	72
Perf Intervals _____	72
Interval Id _____	73
CounterIds and MetricIds _____	73
Starting Time/Ending Time/Maximum Samples _____	73
Configuring Intervals for Statistics _____	74
Creating Performance Intervals _____	74
Updating Performance Intervals _____	74
Remove Performance Intervals _____	74
Querying Statistics _____	76
Retrieving MetricIds _____	76
Querying Statistics _____	76
Querying Information for an Entity and Its Children _____	76
Querying Performance Provider Information _____	77
Querying Metadata Information _____	78
Glossary _____	79
Performance Counters in 1.x and 2.0 _____	89
Revision History _____	93
Index _____	95

An Overview to Porting

This *VMware Infrastructure SDK Porting Guide* contains the tasks which are most frequently ported from an SDK 1.x legacy system to SDK 2.0. Each task contains a description of the differences between SDK 1.x and SDK 2.0, as these differences relate to the code. This information should give you an idea of how you should modify your SDK 1.x code to perform the same task in SDK 2.0.

Developers who use this guide should be familiar with the operation and management of VMware® VirtualCenter, VMware ESX Server, VMware GSX Server, and other VMware products.

Overview of the Porting Guide

This porting guide discusses the most common tasks you encounter:

- [Handling Inventory on page 17](#)

This chapter describes how the tasks associated with inventory differ between SDK 1.x and SDK 2.0. This chapter includes:

 - [Getting Inventory Contents on page 18](#)
 - [Getting Updates on Inventory Contents on page 26](#)
 - [Creating Inventory on page 28](#)
 - [Moving Objects in Inventory on page 33](#)
 - [Deleting Inventory on page 35](#)
- [Virtual Machines and Hosts on page 37](#)

This chapter describes how the tasks associated with virtual machines and hosts differ between SDK 1.x and SDK 2.0. This chapter includes:

 - [Creating a Virtual Machine on page 38](#)
 - [Creating a Virtual Disk on page 43](#)
 - [Provisioning a Virtual Machine on page 44](#)
 - [Updating Configurations on page 46](#)
 - [Connecting and Disconnecting Hosts on page 48](#)
 - [Migrating Virtual Machines on page 50](#)
 - [Deleting Hosts and Virtual Machines on page 54](#)
- [Power Operations on page 55](#)

This chapter describes how the tasks associated with powering on or off differ between SDK 1.x and SDK 2.0. This chapter includes:

 - [Powering On a Virtual Machine on page 56](#)
 - [Powering Off a Virtual Machine on page 57](#)
 - [Suspending a Virtual Machine on page 59](#)
 - [Resetting a Virtual Machine on page 60](#)

- [Event Messages on page 61](#)

This chapter describes how event messages are formatted between SDK 1.x and SDK 2.0. This chapter includes:

- [Mapping Events in SDK 1.x and SDK 2.0 on page 62](#)
 - [Formatting Event Messages on page 68](#)
- [Performance Monitoring on page 71](#)

This chapter describes how the tasks associated with performance monitoring differ between SDK 1.x and SDK 2.0. This chapter includes:

- [Performance Monitoring in SDK 1.x and 2.0 on page 72](#)
- [Configuring Intervals for Statistics on page 74](#)
- [Querying Statistics on page 76](#)
- [Querying Metadata Information on page 78](#)

Using This Porting Guide

This *VMware Infrastructure SDK Porting Guide* describes the differences between SDK 1.x and 2.0, and provides detailed samples illustrating how to port your SDK 1.x client applications to SDK 2.0 client applications.

Note: The *VMware Infrastructure SDK Documentation Roadmap* provides a guide to usage of the documentation included with the VMware Infrastructure SDK.

This *VMware Infrastructure SDK Getting Started Guide* contains a description of the data models, or logical structure of the Virtual Infrastructure Web Service.

The *VMware Infrastructure SDK Reference Guide* contains a complete list of the managed object types and data object types comprising the VMware Infrastructure SDK object model. It lists all the properties, methods, enumerations, and faults present in `vim.wsdl`.

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

Intended Audience

This porting guide is written for programmers who 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 Infrastructure SDK supports VMware VirtualCenter 1.2 and 2.0, and ESX Server 2.0.1, 2.1.x, 2.5.x, and 3.0, and GSX Server 3.1 and 3.2.

Managed Objects and Data Objects

There are two kinds of objects in the VMware Infrastructure SDK: managed objects and data objects. Both are instances of composite object types, but there are important differences.

- Managed object types are not present in the WSDL schema, and are treated in a black box fashion by clients.
- Managed objects exist only on the server, and are passed by reference in the WSDL data stream. The term “managed object reference” denotes a reference to a server-side object that can be accessed only indirectly by the client.
- Data objects can be serialized into the WSDL data stream. In effect, this allows them to be passed by value between the client and the Web service.

For more information about managed objects and data objects, refer to the *VMware Infrastructure SDK Getting Started Guide*.

Managed Object References: Accessing Managed Objects

Managed object types are never passed in their entirety between the client and the server. Either the client works with a reference to a managed object, or the client works with data objects that represent parts of the managed object.

The operations described in this guide require a managed object reference as the first argument. There are several ways to obtain a managed object reference:

- Construct a `ManagedObjectReference` (ServiceInstance managed object only).

You do this as follows (as shown in [Logging On on page 13](#)):

```
ManagedObjectReference sir = new ManagedObjectReference();
sir.setType("ServiceInstance");
sir.set_value("ServiceInstance");
```

- Obtain a reference through the `ServiceContent` data object.

Some references you can obtain through the properties of the `ServiceContent` data object. For example, when you are getting inventory contents (as described in [Getting Inventory Contents on page 18](#)), you never construct a reference to the `PropertyCollector` managed object. Instead, you get a reference in two steps:

- Invoke the `RetrieveServiceContent` operation using the reference to the `ServiceInstance` managed object.

```
ServiceContent sic = vimService.retrieveServiceContent(sir);
```

This returns the `ServiceContent` data object type, which contains a number of managed object references among its properties, one of which is the `PropertyCollector`.

- b. Use the PropertyCollector property of the ServiceContent to get a reference to a PropertyCollector managed object.

```
ManagedObjectReference pcRef = sic.getPropertyCollector();
```

- Invoke an operation that returns a reference to a managed object.

For example, the CreateFilter operation CreateFilter operation (used in [Getting Updates on Inventory Contents on page 26](#)) returns a reference to a PropertyFilter managed object.

```
ManagedObjectReference pfRef = createFilter(pcRef, pfSpec, true);
```

Where to Find SDK 1.x Equivalents

The following table shows an alphabetical list of SDK 1.x objects and the section in this guide that explains their equivalents.

SDK 1.x	Section in this Porting Guide
CloneVM	Cloning and Deploying Virtual Machines on page 40
Create	Creating a Virtual Machine from Scratch on page 38, Creating Inventory on page 28
CreateTemplate	Creating a Template on page 41
CreateVirtualDisk	Creating a Virtual Disk on page 43
Delete	Deleting Inventory on page 35
DisableHost	Disconnecting the Host from VirtualCenter on page 49
EnableHost	Connecting the Host to VirtualCenter on page 48
EventDecl	Formatting Event Messages on page 68
GetContents	Updating Configurations on page 46
GetUpdates	Getting Updates on Inventory Contents on page 26
MigrateVM	Migrating Virtual Machines on page 50
MoveVM	Migrating Virtual Machines on page 50
PerfCollector	Performance Monitoring in SDK 1.x and 2.0 on page 72
PutUpdates	Updating Configurations on page 46
Rename	Moving Objects in Inventory on page 33
ResetVM	Resetting a Virtual Machine on page 60
ResolvePath	Handling Inventory on page 17
StartVM	Powering On a Virtual Machine on page 56
StopVM	Powering Off a Virtual Machine on page 57
VirtualMachineSpec	Provisioning a Virtual Machine on page 44

Before Porting Your SDK 1.x Client Applications

Before porting your SDK 1.x applications, you need to understand the changes made to the SDK architecture in version 2.0. See the *VMware Infrastructure SDK Getting Started Guide* for complete information about the architecture. You should read this guide thoroughly before attempting to port your client applications.

Note: Be sure to read the *VMware Infrastructure SDK Getting Started Guide*. We assume that you are familiar with the terms and concepts introduced in that guide. Consequently, the information presented here summarizes only the essential information required for porting your client applications.

Permissions

Make sure you have the correct permissions for the SDK. See the *VMware Infrastructure SDK Getting Started Guide* for a complete description of the permissions needed.

Logging On

Before your client applications can perform any of the tasks described in this guide, they must first log on. The code described in this section is normally the first snippet of code in your application.

Logging On Using SDK 1.x

In SDK 1.x, you constructed a `VmaServiceLocator` type to define the URL for the Web service and the port to call the service. You also set the a boolean which enabled cookies for the application. Once you had these lines of code in place, you could log on.

The following sample shows the code in SDK 1.x for logging on:

```
VmaService vmaservice = new VmaServiceLocator();
VmaPortType vmaPort = new VmaBindingStub(new URL(args[0]), vmaservice);
((Stub) vmaPort)._setProperty(
    Stub.SESSION_MAINTAIN_PROPERTY,
    Boolean.TRUE);

// Logs on to the server using the login() method.
vmaPort.login(args[1], args[2]);
```

Logging On Using SDK 2.0

In SDK 2.0, the process is nearly the same, except that you must reference two managed objects: `ServiceInstance` and `SessionManager`. The service instance is the central access point for all management data. It is roughly equivalent to the Server Farm found in earlier versions of the SDK. The `ServiceInstance` managed object provides the `RetrieveServiceContent` operation which returns the `ServiceContent` data object. This data object contains various managed object references. The `SessionManager` managed object enables you to log on.

The following sample code shows the logon information for SDK 2.0.

```
/*
   Set the url and turn cookies on. The URL is passed in as the
   first command line argument of the client application. VimServiceLocator
   and VimPortType are two of the stubs generated by the SDK toolkit.
*/
URL url = new URL(args[0]);
VimServiceLocator vim = new VimServiceLocator();
vim.setMaintainSession(true);
VimPortType vimService = vim.getVimPort(url);

/*
   Create a managed object reference to ServiceInstance. The Type,
   "service instance", is one of the valid values for the Type property as
   listed in the VMware Infrastructure SDK Reference Guide.
*/
```

```
ManagedObjectReference sir = new ManagedObjectReference();
sir.setType("ServiceInstance");
sir.set_value("ServiceInstance");

/*
   Invokes the RetrieveServiceContent method. This method returns the
   ServiceContent data object type. This gives access to the
   sessionManager property, the managed object for logging in and managing
   sessions.
*/
ServiceContent sic = vimService.retrieveServiceContent(sir);

/*
   In the following code, the SessionManager Reference is assigned to
   the SessionManager managed object reference, and the Login method is
   invoked. The Login method takes as its arguments the session manager
   reference (smr), two arguments passed in from the command line
   (the userid and the password), and the locale. Here the
   locale is defined as "null". For more information about the locale
   parameter, see the Login method of the SessionManager managed
   object type in the VMware Infrastructure SDK Reference Guide.
*/
ManagedObjectReference smr = sic.getSessionManager();
UserSession us = vimService.login(smr, args[1], args[2], null);
```

Technical Support Resources

Standards and Reference Documentation

Refer to the following Web sites for additional information.

- VMware Infrastructure 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

Handling Inventory

This chapter describes the differences between SDK 1.x and SDK 2.0 when you are creating client applications for handling inventory objects. The chapter includes:

- [Getting Inventory Contents on page 18](#)
- [Getting Updates on Inventory Contents on page 26](#)
- [Creating Inventory on page 28](#)
- [Moving Objects in Inventory on page 33](#)
- [Renaming Objects in Inventory on page 34](#)
- [Deleting Inventory on page 35](#)

Getting Inventory Contents

To perform most operations, you need to identify the managed object on which the operation will be performed. You often need to find certain properties of the managed object as well (for example, when you want to get the runtime information on a virtual machine).

Getting Contents Using SDK 1.x

In SDK 1.x, you used the `ResolvePath` operation to find a virtual machine. Once you had the handle, you used the `GetContents` operation.

```
String path = "/vm/<New Farm Group>/<New Farm>/<Fully qualified vmName>"
ViewContents vc = null;
try {
    // serviceConnection is com.vmware.vma.VmaPortType
    String vmHandle = serviceConnection.resolvePath(path);
    vc = serviceConnection.getContents(vmHandle);
}
catch (Exception ex) {
    System.out.println("Got Exception calling getContents : " +
        ex.getMessage());
    ex.printStackTrace(System.out);
    throw ex;
}
```

This returned all the information for the virtual machine at the location represented by `vmPath`.

Getting Contents Using SDK 2.0

In SDK 2.0, instead of using `GetContents` to retrieve all the properties for a managed object, you create a `PropertyFilterSpec` object.

The `PropertyFilterSpec` Data Object

A `PropertyFilterSpec` defines a filter for searching through a hierarchy of managed objects. You can define a `PropertyFilterSpec` that looks for any managed objects and their properties, including `ManagedEntity` objects. For example, as described in [Using a PropertyFilterSpec to Find Information about Tasks on page 24](#), you can retrieve the `recentTask` property of the `TaskManager` managed object to identify recent tasks. The `PropertyFilterSpec` consists of some or all of the following, depending on what you are seeking:

- `PropertySpec` — Defines the managed object type to select as well as any properties belonging to that managed object.
- `ObjectSpec` — Defines the managed object that will be the starting point for the search.

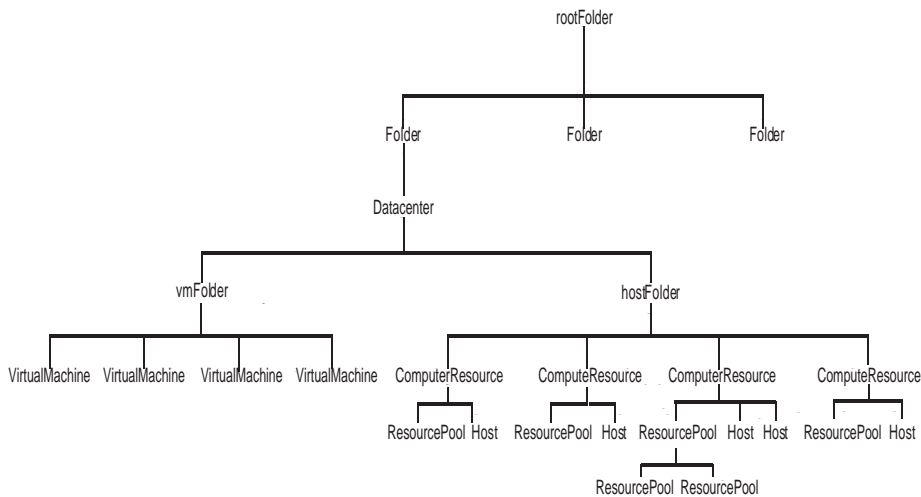
- TraversalSpec — Defines a rule that will be fired if a particular managed object is encountered. The TraversalSpec identifies the managed object as well as a property of the managed object to which the search will traverse.
- SelectionSpec — Selects a TraversalSpec rule to be fired.

To initiate the search and retrieve the results, you invoke a RetrieveProperties operation using this PropertyFilterSpec object as well as a reference to a PropertyCollector managed object.

This section describes two types of PropertyFilterSpecs. One starts with a Folder managed object and traverses a hierarchy of ManagedEntity objects to find two properties of VirtualMachine objects. The other describes using a PropertyFilterSpec to find information about Task managed objects.

Using a PropertyFilterSpec to Search for VirtualMachines

In this section, a PropertyFilterSpec is used to search through a hierarchy of ManagedEntity objects to find config and runtime properties of the summary property of VirtualMachine objects. The object uses the following hierarchy of managed objects.



The sample code snippet begins at a starting object in the inventory tree shown above (for example, rootFolder) and, using a series of TraversalSpec objects as rules, fires the rules at each managed object node until there are no more nodes in the hierarchy that meet the rules.

```

public PropertyFilterSpec[] createPFSForVMs(ManagedObjectReference folderRef)
{

```

```

// First define what data we want back from the filter
// by creating a PropertySpec
PropertySpec pSpec = new PropertySpec();
pSpec.setAll(Boolean.FALSE);
pSpec.setType("VirtualMachine");
pSpec.setPathSet(new String[]{"summary.config", "summary.runtime"});

// Now define how to find that data by creating an ObjectSpec
ObjectSpec oSpec = new ObjectSpec();
oSpec.setObj(folderRef);
oSpec.setSkip(Boolean.TRUE);

// Set how we want to select our data by creating a TraversalSpec
TraversalSpec folderSpec = new TraversalSpec();
String recurseFolders = "recurseFolders";
folderSpec.setName(recurseFolders);

// Set the type of object we want to use. This MUST match the
// type of object in setObj() above if this TraversalSpec is
// the 'top level' TraversalSpec
folderSpec.setType("Folder");

// Set the property to use for getting objects to filter
// 'childEntity' returns a ManagedEntity[] of children of the Folder
folderSpec.setPath("childEntity");

// Make sure the results from this traversal are considered by
// the filter specified in the PropertySpec
folderSpec.setSkip(Boolean.FALSE);

// Create a SelectionSpec to refer to the top-level TraversalSpec so
// we can recurse down the tree of objects
SelectionSpec recurseSpec = new SelectionSpec();
recurseSpec.setName(recurseFolders);

// Create a TraversalSpec to traverse through a Datacenter using
// it's vmFolder property
TraversalSpec datacenterSpec = new TraversalSpec();
datacenterSpec.setType("Datacenter");
datacenterSpec.setPath("vmFolder");

// Set skip to true because we don't want information from Folders
datacenterSpec.setSkip(Boolean.TRUE);

// Add the recurse SelectionSpec so we'll recurse down the Folder
datacenterSpec.setSelectSet(new SelectionSpec[]{recurseSpec});

```

```

// Associate the Recurse SelectionSpec and the
// Datacenter TraversalSpec with the Folder TraversalSpec
folderSpec.setSelectSet(new SelectionSpec[] {recurseSpec,
    datacenterSpec});

// Associate the TraversalSpec with the ObjectSpec
oSpec.setSelectSet(new SelectionSpec[] {folderSpec});

// Now put the whole thing together in the PropertyFilterSpec
PropertyFilterSpec pfSpec = new PropertyFilterSpec();
pfSpec.setPropSet(new PropertySpec[] {pSpec});
pfSpec.setObjectSet(new ObjectSpec[] {oSpec});

return new PropertyFilterSpec[] {pfSpec};
}

```

Building a PropertyFilterSpec to Find VirtualMachine Objects

Using the inventory tree shown above, let's say that you want to filter for summary configuration and summary runtime information for all virtual machines.

1. Construct the PropertyFilterSpec and set its properties.

```

PropertyFilterSpec pfSpec = new PropertyFilterSpec();
pfSpec.setPropSet(new PropertySpec[] {pSpec});
pfSpec.setObjectSet(new ObjectSpec[] {oSpec});

```

The propSet property defines the PropertySpec to be used for the filter. The PropertySpec defines the object type for which you are searching as well as the properties (if any) belonging to that object.

The objectSet property defines the ObjectSpec to be used for the filter. The ObjectSpec defines where the filter begins as well as how the filter proceeds from there.

2. Construct the ObjectSpec and set its properties.

```

ObjectSpec oSpec = new ObjectSpec();
oSpec.setObj(folderRef);
oSpec.setSkip(Boolean.TRUE);
oSpec.setSelectSet(new SelectionSpec[] {folderSpec});

```

The ObjectSpec object defines the starting point for the filter (the Obj property). This starting point can be either a location in the inventory tree (the rooFolder, DataCenter, and so forth) or it could be the managed object type that has the properties you want to get (for example, Task).

In this case, the starting object is a reference to a Folder managed object passed in as a parameter to the PropertyFilterSpec. The advantage is that you can set your starting object

wherever is most efficient for the search. If you are searching for virtualmachines, for example, starting at /rootFolder is inefficient since a search from there wastes time searching nodes that have no VirtualMachine objects in them (any Folders that are parent to a Datacenter). It is more efficient to start searching at a vmFolder.

If the skip boolean is set to true, then the filter does not check to see if the starting object's type matches any of the types listed in the associated PropertySpec.

The selectSet property defines what should be done at the starting point. This is defined with a SelectionSpec. This code constructs a SelectionSpec[] from a variable, folderSpec, that is defined later.

3. Construct the PropertySpec and set its properties.

```
PropertySpec pSpec = new PropertySpec();
pSpec.setAll( Boolean.FALSE );
pSpec.setType( "VirtualMachine" );
pSpec.setPathSet( new String[] { "summary.config", "summary.runtime" } );
```

The Type property determines the type of managed object being sought (in this case, a VirtualMachine). The pathSet property determines the properties of the managed object, in this case, the config property of the summary property and the runtime property of the summary property.

The Type property must match a managed object type. The valid values are listed in the Type property of the ManagedObjectReference data object type in the *VMware Infrastructure SDK Reference Guide*. For example, if you are looking for virtual machines, then the string must be "VirtualMachine". If the managed object is a host system, then the value of the Type property must be "HostSystem".

The setting of the All boolean determines whether you are seeking all the properties of the managed object or only the subset defined by the pathSet property.

- If you set the All boolean to true, then the collector filters for all the properties belonging to the object identified by the Type property.
- If you set the All boolean to false and do not provide a pathSet value, then the collector retrieves only the object reference of the Type property. (The default setting of the All boolean is false.)
- If you set the All boolean to false and provide one or more values for the pathSet property, then the collector retrieves the properties represented by pathSet.

4. Construct the TraversalSpec referred to by the ObjectSpec and set its properties.

```
TraversalSpec folderSpec = new TraversalSpec();
String recurseFolders = "recurseFolders";
folderSpec.setName( recurseFolders );
```

```

folderSpec.setType("Folder");
folderSpec.setPath("childEntity");
folderSpec.setSkip(Boolean.FALSE);
folderSpec.setSelectSet(new SelectionSpec[] {recurseSpec,
    datacenterSpec});

```

TraversalSpec objects define what is done when the filter encounters a certain type of managed object type. In this example, the Type property defines the managed object (Folder). The Path property defines the property of the node (childEntity) that is sought. In this case, the rule says that if the current object is a Folder, for each childEntity:

- a. Set the current object.
- b. Apply the PropertySpec and look for matches to the object (Type) set there.

The selectSet property defines the SelectionSpec object. The SelectionSpec object defines additional actions to take. In this case, the recurseSpec and dataCenterSpec variables refer to variables to be created in the next few steps. These variables repeat this TraversalSpec, then fire a new TraversalSpec to search the Datacenter object.

5. Construct the SelectionSpec needed for the first TraversalSpec.

```

// Create a SelectionSpec to refer to the top-level TraversalSpec so
// we can recurse down the tree of objects
SelectionSpec recurseSpec = new SelectionSpec();
recurseSpec.setName(recurseFolders);

```

Notice that the name, `recurseFolders`, matches the value of the selectSet property in Step 4. This means that this SelectionSpec fires the TraversalSpec with that name.

6. Construct the TraversalSpec needed for Step 4.

In general, you need a TraversalSpec for each node that you want to traverse. Given the managed object hierarchy shown earlier, if you are filtering for VirtualMachine object types, you need a TraversalSpec that traverses vmFolders of Datacenter object types.

```

TraversalSpec datacenterSpec = new TraversalSpec();
datacenterSpec.setType("Datacenter");
datacenterSpec.setPath("vmFolder");
datacenterSpec.setSkip(Boolean.TRUE);
datacenterSpec.setSelectSet(new SelectionSpec[] {recurseSpec});

```

In this case, the rule says that if the current object is a Datacenter, for each vmFolder:

- a. Set the current object.
- b. Apply the PropertySpec and look for matches.

The `selectSet` property defines the `SelectionSpec` object which defines additional actions to take. In this case, the `recurseSpec` variable refers to the `SelectionSpec` defined in Step 5. This fires the initial `TraversalSpec` (`folderSpec`).

How the Selection Process Works

These `SelectionSpec` objects associated with the `ObjectSpec` specify the following rules starting with the starting object defined in the `ObjectSpec` (`objSpec[0].setObj(startObject)`).

1. If the current object is a `Folder`, for each element of the `childEntity` property:
 - a. Select the current object to read its properties.
 - b. Apply the `PropertySpec` and look for matches.
 - c. Repeat this rule.
2. If the current object is a `Datacenter`, for each `vmFolder` (`Type = Folder`):
 - a. Select the current object to read its properties.
 - b. Apply the `PropertySpec` and look for matches.
 - c. Repeat this rule.

Using The RetrieveProperties Operation to Retrieve the Filtered Information

After you have created the `PropertyFilterSpec` for the selection, you invoke the `RetrieveProperties` operation to get the results.

```
ManagedObjectReference pcRef = sic.getPropertyCollector();
ObjectContent[] objContent = vmService.retrieveProperties(pcRef, pfs);
```

The ObjectContent Data Object

The `RetrieveProperties` operation returns an array of `ObjectContent` objects. Each `ObjectContent` object contains the reference to the managed object that provided the properties retrieved as well as an array of `DynamicProperty` objects. Each `DynamicProperty` object contains the name and value of the property. The value of the property could be an array that needs to be handled specially.

Using a PropertyFilterSpec to Find Information about Tasks

Sometimes you want to search for information about managed objects other than `ManagedEntity` objects. For example, you might want to retrieve information about `Task` managed objects. In this example, we are trying to find tasks that completed recently, are currently running, or are queued to run. One way to do this is to find the property that has this information (`recentTask`), then find its parent managed object type (in this case, `TaskManager`). This is your starting object. The managed object type you are looking for is `Task`. You define the `pathSet` property of the `PropertySpec` as the properties of the managed object you want.

The hierarchy is simple:



The sample code snippet is as follows:

```

public PropertyFilterSpec[] createPFForRecentTasks(ManagedObjectReference
taskManagerRef) {

    PropertySpec pSpec = new PropertySpec();
    pSpec.setAll(Boolean.FALSE);
    pSpec.setType("Task");
    pSpec.setPathSet(new String[]{"info.state", "info.cancelled",
        "info.error"});

    // Now define how to find that data by creating an ObjectSpec.
    ObjectSpec oSpec = new ObjectSpec();
    oSpec.setObj(taskManagerRef);

    // Don't want to skip the TaskManager object since that has the data!
    oSpec.setSkip(Boolean.FALSE);

    TraversalSpec tSpec = new TraversalSpec();
    tSpec.setType("TaskManager");
    tSpec.setPath("recentTask");

    // Make sure the results from this traversal are considered by the
    // filter specified in the PropertySpec.
    tSpec.setSkip(Boolean.FALSE);

    // Associate the TraversalSpec(s) with the ObjectSpec.
    oSpec.setSelectSet(new SelectionSpec[]{tSpec});

    // Now put the whole thing together in the PropertyFilterSpec.
    PropertyFilterSpec pfSpec = new PropertyFilterSpec();
    pfSpec.setPropSet(new PropertySpec[]{pSpec});
    pfSpec.setObjectSet(new ObjectSpec[]{oSpec});

    // Now return the PropertyFilterSpec for use.
    return new PropertyFilterSpec[]{pfSpec};
}
  
```

Getting Updates on Inventory Contents

When you are getting updates on inventory contents, SDK 2.0 differs from SDK 1.x in using property filters to define the objects and properties for which you want to get updates.

Getting Updates in SDK 1.x

In SDK 1.x, once you found the object (using the `ResolvePath` operation) for which you wanted updates, you used one of two operations: `GetContents` or `GetUpdates`. `GetContents` returned the entire version of the inventory object. `GetUpdates` returned only the changes.

The following sample code shows the use of the `GetUpdates` operation:

```
ViewContents vc = serviceConnection.resolvePath("/vm/564d5a05-29a7-b09b-d576-9cb8a719d940");
VirtualMachine clientData = (VirtualMachine) (vc.getBody());
boolean wait = true;
while (true)
{
    VHandleList vHandleList = new VHandleList();
    vHandleList.setVHandle(new String[] { vc.getVHandle() });
    UpdateList updateList = serviceConnection.getUpdates(vHandleList, wait);
    Update [] updates = updateList.getUpdate();
    // Since we've requested updates only on 1 vHandle, updates.length will be 1
    for (int i = 0; i < updates.length; i++)
    {
        String handle = updates[i].getHandle();
        Change[] changes = updates[i].getChange();
        for (int j = 0; j < changes.length; j++)
        {
            //... apply change to client data ...
            processChange(handle, changes[j]);
        }
        vc.setVHandle(updates[i].getVHandle());
    }
}
```

Getting Updates in SDK 2.0

In SDK 2.0, instead of `GetUpdates`, you use one of two operations: `CheckForUpdates` or `WaitForUpdates`. The `CheckForUpdates` operation returns the changes since the last updated version of the property. If no updates are pending, then the operation returns null. The `WaitForUpdate` operation returns changes since the last updated version. If there are no changes, however, then this operation waits until updates are completed.

In SDK 2.0, getting updates on inventory contents requires you to do the following:

1. Construct a PropertyFilterSpec object.

The PropertyFilterSpec object filters for the inventory object (or objects) and the properties whose updates you want. See [Getting Inventory Contents on page 18](#) for complete information about the PropertyFilterSpec object.

2. Obtain a reference to a PropertyCollector managed object by invoking the RetrieveServiceContent operation.

The RetrieveServiceContent operation returns a ServiceContent data object type. The PropertyCollector managed object reference is returned using an accessor method on the propertyCollector property:

```
ManagedObjectReference sir = new ManagedObjectReference();
sir.setType("ServiceInstance");
sir.setValue("ServiceInstance");

ServiceContent sic = retrieveServiceContent(sir);
ManagedObjectReference pcRef = sic.getPropertyCollector();
```

3. Create a property filter by invoking the CreateFilter operation.

The operation takes three parameters: the PropertyCollector managed object reference, the PropertyFilterSpec, and a boolean partialUpdates. The partialUpdates boolean determines whether or not changes to nested objects are reported. If set to true, then all levels are reported.

```
ManagedObjectReference pfRef = createFilter(pcRef, pfspec, true);
```

4. Invoke the CheckForUpdates or WaitForUpdates operation with the PropertyCollector managed object reference as the parameter.

Creating Inventory

Creating inventory in SDK 2.0 differs from SDK 1.x in the following ways:

- SDK 2.0 requires you to create a ManagedObjectReference for a Folder managed object.
- Some types of SDK 1.x inventory objects equate to different objects in SDK 2.0, as shown in the following table:

SDK 1.x	SDK 2.0
Container, VirtualMachineGroup	Folder
Farm	Datacenter

- Because of new functionality in SDK 2.0, you can create objects not found in SDK 1.x. You can create a ClusterComputeResource managed object. Because of this, you can create two types of Host objects: the standalone host and the clustered host.
- Some types of objects you could create in SDK 1.x are created differently in SDK 2.0. In SDK 1.x, you create a TaskSchedule or PerfCollector object directly. In SDK 2.0, these objects are created automatically when you perform operations (for example, a power operation).

Creating Inventory in SDK 1.x

In the following sample code snippet, the parentPath was the location in the tree where you want to create the object. The serviceConnection argument was created during the Login process. See [Logging On on page 13](#). When you created a Host object, you specified the initial value of the Host object with a HostSpec.

```
public static void createElement(
    String parentPath,
    String name,
    String type,
    VmaPortType serviceConnection) {
    try {
        Object initVal = null;
        // Get handle for parent node
        String parentHandle = serviceConnection.resolvePath(parentPath);

        if (type.equals("Container"))
            initVal = (Object) new com.vmware.vma.Container();
        else if (type.equals("Farm"))
            initVal = (Object) new com.vmware.vma.Farm();
        else if (type.equals("VirtualMachineGroup"))
            initVal = (Object) new com.vmware.vma.VirtualMachineGroup();
        else if (type.equals("Host")) {
            HostSpec hostSpec = new com.vmware.vma.HostSpec();
```

```

        // Prompt user for username & password for this host
        String userName = getUserName();
        String password = getPassword();
        Integer port = getPort();

        hostSpec.setPort(port);
        hostSpec.setUserName(userName);
        hostSpec.setPassword(password);
        initVal = hostSpec;
    } else if (type.equals("VirtualMachine")) {
        System.out.println(
            "Please refer to "
                + "com.vmware.sample.CreationDeletion.CreateVM");
        return;
    } else if (type.equals("TaskSchedule")) {
        System.out.println(
            "Please refer to "
                +
                "com.vmware.sample.ScheduledTasks.ScheduledTaskSample");
        return;
    } else if (type.equals("PerfCollector")) {
        System.out.println(
            "Please refer to "
                + "com.vmware.sample.PerfMonitor.PerfSample");
        return;
    } else {
        System.out.println("Uknown Type. Allowed types are:");
        System.out.println(" Container");
        System.out.println(" Farm");
        System.out.println(" Host");
        System.out.println(" VirtualMachineGroup");
        System.out.println(" VirtualMachine");
        System.out.println(" TaskSchedule");
        System.out.println(" PerfCollector");
        return;
    }
    String handle = serviceConnection.create(
        parentHandle, name, type, initVal);
    System.out.println("Object created succesfully. Handle of new
        object:" + handle);
} catch (FaultInfo faultInfo) {
    FaultKind faultKind = faultInfo.getKind();
    System.out.println("Error: " + faultKind.toString());
} catch (Exception ex) {
    ex.printStackTrace(System.out);
}
}
}

```

Creating Inventory in SDK 2.0

In SDK 2.0, as shown in the sample code snippet below, the arguments are the same as for SDK 1.x. The VimPortType object in SDK 2.0 maps to the VmaPortType object in SDK 1.x. Like that object, vimService is created during logon (see [Logging On on page 13](#)).

```
public static void createElement(
    String parentRef,
    String type,
    String name,
    VimPortType vimService)
    throws Exception {
    ...
}
```

Unlike SDK 1.x, you must have a reference to the Folder managed object that is the parent for the created object. This is used as an argument later when you create the object. To find the Folder reference:

1. Construct a PropertyFilterSpec.

See [Getting Inventory Contents on page 18](#) for an explanation of searching with a PropertyFilterSpec in SDK 2.0. Depending on how you construct the object, you can return all Folders or, using the name property, the specific Folder.

2. Get a reference to a PropertyCollector managed object.

```
ManagedObjectReference sir = new ManagedObjectReference();
sir.setType("ServiceInstance");
sir.set_value("ServiceInstance");

ServiceContent sic = vimService.retrieveServiceContent(sir);
ManagedObjectReference pcRef = sic.getPropertyCollector();
```

3. Invoke the RetrieveProperties operation to retrieve the results of the PropertyFilterSpec search.

```
ObjectContent[] objContent = vimService.retrieveProperties(pcRef, pfs);
```

This returns an array consisting of the Folders found. Each element contains the reference to a Folder managed object, the path to the Folder, and the name of the Folder.

4. In the ObjectContent array, find the Folder reference and assign that to a variable.

```
ManagedObjectReference parentRef = objContent[5].obj;
```

Once you have the ManagedObjectReference, a series of conditional statements create the inventory objects with a series of pre-defined operations. As in SDK 1.x, the statements use the

Type argument passed in from the Main method. Notice, however, that the types are different for SDK 2.0.

```

if (type.equals("Folder")) {
    vimService.createFolder(folderMoRef, name);
}
else if (type.equals("Datacenter")) {
    vimService.createDatacenter(folderMoRef, name);
}

```

Because SDK 2.0 allows clustering, you can define a ClusterComputeResource object within which you can add one or more clustered Hosts. Because of this, you also need to configure these clusters with properties such as the FailoverLevel.

```

else if (type.equals("Cluster")) {
    ClusterDasConfigInfo cDas = new ClusterDasConfigInfo();
    cDas.setAdmissionControlEnabled(false);
    cDas.setEnabled(true);
    cDas.setFailoverLevel(1);

    ClusterDrsConfigInfo cDrs = new ClusterDrsConfigInfo();
    cDrs.setDefaultVmBehavior(DrsBehavior.manual);
    cDrs.setEnabled(true);
    cDrs.setVmotionRate(ClusterDrsConfigInfoVmotionRate.aggressive);

    ClusterConfigSpec clusterSpec = new ClusterConfigSpec();
    clusterSpec.setDasConfig(cDas);
    clusterSpec.setDrsConfig(cDrs);
    vimService.createCluster(folderMoRef, name, clusterSpec);
}

```

Because of clustering, you can define two types of hosts: standalone and clustered. The sample code snippet below shows how a conditional statement handles this.

```

else if (type.equals("Host-Standalone") ||
        type.equals("Host-Clustered")) {
    HostConnectSpec hostSpec = new HostConnectSpec();
    // Prompt user for username & password for this host
    hostSpec.setHostName(name);
    hostSpec.setUserName(getUserName());
    hostSpec.setPassword(getPassword());
    hostSpec.setPort(getPort());
    if (type.equals("Host-Standalone")) {
        vimService.addStandaloneHost_Task(folderMoRef, hostSpec, true);
    }
    else {
        ManagedObjectReference ccrRef = new ManagedObjectReference();

```

```
        ccrRef.setType("ClusterComputeResource");  
        ccrRef.set_value(parentRef);  
        vimService.addHost_Task(ccrRef, hostSpec, true);  
    }  
}
```

A final sample code snippet handles any unknown types the user might enter.

```
    } else {  
        System.out.println("Unknown Type. Allowed types are:");  
        System.out.println(" Host-Standalone");  
        System.out.println(" Host-Clustered");  
        System.out.println(" Cluster");  
        System.out.println(" Datacenter");  
        System.out.println(" Folder");  
    }  
}
```

Moving Objects in Inventory

In SDK 1.x, you moved and renamed objects such as Hosts, Virtual Machines, and Folders using a single set of code. In SDK 2.0, you move and rename objects using two separate operations.

Moving Objects In SDK 1.x

In SDK 1.x, you moved and renamed an object using the Rename operation. The renaming was handled with an optional name parameter. If the arguments did not include a name parameter, the object was moved and the name was unchanged.

Moving Objects In SDK 2.0

In SDK 2.0, you move and rename an object using separate move and rename operations. For a description of the Rename_Task operation, see [Renaming Objects in Inventory on page 34](#).

To move an object, you use the MoveIntoFolder_Task operation. This operation takes two parameters: a reference to the Folder managed object into which you want to move the object and a reference to ManagedEntity managed objects that are being moved. You obtain the references to these objects by using the selection process described in [Getting Inventory Contents on page 18](#).

```
vimService.moveIntoFolder_Task(destFolderRef,  
    new ManagedObjectReference[] {meRef});  
System.out.println("Operation successful");
```

Renaming Objects in Inventory

As described in [Moving Objects in Inventory on page 33](#), SDK 1.x used a single application to move and rename objects. In SDK 2.0, you use a separate application to rename an object.

Renaming Objects in SDK 1.x

In SDK 1.x, the command line arguments for the Rename operation included two optional arguments: the destination to which you were moving the object and the new name for the object. If the destination was excluded from the command line arguments, the object was renamed without moving it. If the name was excluded, the object was moved without renaming.

Renaming Objects in SDK 2.0

In SDK 2.0, the Main method takes two additional arguments: A reference to the ManagedEntity object to be renamed and the new name. You obtain the managed object reference by using the process described in [Getting Inventory Contents on page 18](#).

Once you have obtained the managed object reference, you invoke the Rename_Task operation with the managed object reference and the new name.

```
vimService.rename_Task(meRef, newName);  
System.out.println("Operation successful");
```

Deleting Inventory

The code for deleting a host or virtual machine in SDK 2.0 is similar to that in SDK 1.x. As with the other code in this guide, there are two main differences. In SDK 2.0, you create a property filter to find the managed entity on which you wish to perform the action. In addition, in SDK 2.0, before you can invoke methods and get properties, you must create a managed object reference.

SDK 1.x

In SDK 1.x, after finding the inventory object with the ResolvePath operation, you used the Delete operation to delete the object.

The following code snippet shows an object being deleted in SDK 1.x:

```
String handle = serviceConnection.resolvePath(path);
serviceConnection.delete(handle);
System.out.println("Operation successful");
```

SDK 2.0

In SDK 2.0, instead of the Delete operation, you use the Destroy_Task operation. You obtain a reference to the ManagedEntity you want to delete by using the process described in [Getting Inventory Contents on page 18](#). Then you invoke the Destroy_Task operation on the managed object reference you obtained.

```
// Call destroy
vimService.destroy_Task(meRef);
System.out.println("Operation successful");
```


CHAPTER 3

Virtual Machines and Hosts

This chapter describes the differences between SDK 1.x and SDK 2.0 for client applications associated with virtual machines and hosts. The chapter includes:

- [Creating a Virtual Machine on page 38](#)
- [Creating a Virtual Disk on page 43](#)
- [Provisioning a Virtual Machine on page 44](#)
- [Updating Configurations on page 46](#)
- [Connecting and Disconnecting Hosts on page 48](#)
- [Migrating Virtual Machines on page 50](#)
- [Deleting Hosts and Virtual Machines on page 54](#)

Creating a Virtual Machine

You can create a virtual machine using one of three ways.

- Create a virtual machine from scratch.
In SDK 1.x, you used the Create operation. In SDK 2.0, you use CreateVM_Task.
- Clone an existing virtual machine.
In SDK 1.x, you used the CloneVM operation. In SDK 2.0, you used the CloneVM_Task operation, but with different parameters.
- Create a virtual machine from a template.
In SDK 1.x, you used the CloneVM operation. In SDK 2.0, you change a boolean flag to change a template to a virtual machine.

Creating a Virtual Machine from Scratch

In SDK 1.x, a single operation was used to create all objects, including virtual machines. In SDK 2.0, the CreateVM_Task operation is specifically used to create virtual machines.

Creating from Scratch in SDK 1.x

In SDK 1.x, you used a Create operation to create a variety of inventory objects, one of which was a virtual machine. As parameters, you passed in the following:

- `parentHandle` — the 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.
- `name` — The name of the virtual machine in the hierarchy.
- `type` — The type of the new object; in this case, `VirtualMachine`.
- `initial` — The virtual machine configuration spec.

In SDK 1.x, if you did not specify a `parentHandle` parameter, then the operation placed the newly created virtual machine in a default Farm.

A template was a separate entity in SDK 1.x. Therefore, you used a separate operation to create a template. See [Creating a Template on page 41](#).

Creating in SDK 2.0

The CreateVM_Task operation, used specifically to create the virtual machine, takes the following input parameters:

- A reference to a Folder managed object — This is the folder where you want to locate the virtual machine. Unlike in SDK 1.x, there is no default location. There must always be a Folder parameter locating the virtual machine.

- `config` — This is a `VirtualMachineConfigSpec` object and maps to the initial parameter in SDK 1.x. This object contains all the configuration information for the virtual machine, including a boolean `templateOnly` property. In SDK 2.0, a template is not a separate entity, so you do not need a separate operation to create a template. When you create a virtual machine, you use this boolean property to designate the virtual machine as a template. See [Creating a Template on page 41](#).
- `pool` — A reference to a `ResourcePool` managed object. It represents the resource pool to which the virtual machine should be attached. You obtain this managed object reference using the process described in [Handling Inventory on page 17](#).
- `host` — A reference to a `HostSystem` managed object. It represents the target host on which to run the virtual machine. You obtain this managed object reference using the process described in [Handling Inventory on page 17](#). This must specify a host that is a member of the `ComputeResource` indirectly specified by the `pool` argument. For a stand-alone host or a cluster with DRS, this parameter can be omitted, and the system selects a default.

The following sample code snippet creates a virtual machine. The managed object references are obtained with user-defined helper functions (`getDatacenter`, `getHostFolder`, and so on) which point to a `PropertyFilterSpec` for retrieving these references. For the complete sample code, refer to the sample code located in `/SDK/sample_2.0/Axis/java/com/vmware`.

```

...
ManagedObjectReference dcmor = _vmUtils.getDatacenter(getDatacenterFolder());
ManagedObjectReference hostfoldermor = _vmUtils.getHostFolder(dcmor);
ManagedObjectReference compresmor =
    _vmUtils.getComputeResource(hostfoldermor);
ManagedObjectReference hostmor = _vmUtils.getHost(hostfoldermor,
    getCreateOnHost());
ManagedObjectReference resourcePool = _vmUtils.getResourcePool(compresmor);
ManagedObjectReference vmFolderMor = _vmUtils.getVmFolder(dcmor);

VirtualMachineConfigSpec vmConfigSpec =
    _vmUtils.createVmConfigSpec(getVmName(), null, compresmor, hostmor);

vmConfigSpec.setName(getVmName());
vmConfigSpec.setAnnotation(getAnnotation());
vmConfigSpec.setMemoryMB(new Long(getMemorySizeMB()));
vmConfigSpec.setNumCPUs(new Integer(getCpuCount()));
vmConfigSpec.setGuestId(getGuestOsId());

ManagedObjectReference taskmor = vimService.createVM_Task(
    vmFolderMor, vmConfigSpec, resourcePool, hostmor
);
...

```

Cloning and Deploying Virtual Machines

In SDK 1.x, you use CloneVM. In SDK 2.0, you use CloneVM_Task. The parameters differ.

Cloning and Deploying in SDK 1.x

In SDK 1.x, you used the CloneVM with the following parameters:

- srcHandle — The handle to the source Virtual Machine or Template.
- parentHandle — The handle to the destination Farm or virtual machine group in which the cloned virtual machine will be created.
- destHostHandle — The handle to the host in which the cloned virtual machine will reside.
- name — The name of the newly cloned virtual virtual machine.
- datastore — The location on the destination host where the cloned virtual machine's configuration files and virtual disks will reside.
- customization — The configuration document for the newly cloned virtual machine.
- autpoweron — (Optional) A flag that determines whether or not the newly cloned virtual machine automatically powers on once the cloning operation is complete. If the flag is set to true, then the virtual machine powers on automatically.

Cloning and Deploying in SDK 2.0

SDK 2.0 uses CloneVM_Task with the following parameters:

- A reference to a VirtualMachine managed object — This is the source virtual machine for the clone. This maps to the SDK 1.x srcHandle parameter.

Note: The source virtual machine can be either an active virtual machine or a template, depending on the template setting in the configuration information of the source virtual machine. See [Creating a Template on page 41](#) for more information.
- folder — The location of the new virtual machine. This is a reference to a Folder managed object and maps to the parentHandle parameter in SDK 1.x.
- name — As in SDK 1.x, this is the name of the newly cloned virtual machine.
- spec — A VirtualMachineCloneSpec object that specifies how to clone the virtual machine. This object contains properties that define the virtual machine:
 - Whether or not the new virtual machine is a template. This is new. To do this in SDK 1.x would have required that you create the virtual machine (with CloneVM), then create the template (with CreateTemplate). See [Creating a Template on page 41](#) for more information.
 - The location. This VirtualMachineRelocationSpec object defines the datastore location for the virtual machine and the target host. In SDK 2.0, you can locate the virtual disks in separate datastore locations, so this spec also includes a property that specifies the

datastore location for each virtual disk. The properties in this object map to the `destHostHandle` and the `datastore` parameters in SDK 1.x.

- Whether or not to power on (deploy) the virtual machine once it's created. This property maps to the `autopoweron` boolean parameter in SDK 1.x.
- Any customization, including encryption keys, network identities, and so on. This property maps to the `customization` input parameter in SDK 1.x.

Creating a Template

In SDK 1.x, a template was a separate entity from a virtual machine. In SDK 2.0, a template is not a separate entity. In SDK 2.0, a template is a virtual machine defined as a template.

Creating a Template in SDK 1.x

In SDK 1.x, since a template was a separate entity from a virtual machine, you created a template by invoking a `CreateTemplate` operation with two arguments: the handle to the source virtual machine and a `TemplateSpec` that defined the information for the template.

Creating a Template in SDK 2.0

In SDK 2.0, a template is not a separate entity, but rather a quality of a virtual machine. A virtual machine can be either an active virtual machine or a template. There are several ways to create a template in SDK 2.0:

- Create the virtual machine as a template.
Use the `CreateVM_Task` operation (see [Creating a Virtual Machine from Scratch on page 38](#)) to create the virtual machine. One of its parameters, `config`, is a `VirtualMachineConfigSpec` object. This object contains a boolean property called `templateOnly` which, when set to true, designates the new virtual machine as a template.
- Clone a virtual machine as a template.
Use the `CloneVM_Task` operation (see [Cloning and Deploying Virtual Machines on page 40](#)) to clone an existing virtual machine. One of its parameters, `spec`, is a `VirtualMachineCloneSpec` object. This object contains a boolean property called `template` which, when set to true, designates the clone as a template.
- Change a virtual machine to a template.
You can use one of two operations to change an existing virtual machine to a template: `ReconfigVM_Task` or `MarkTemplateAsVM`.
 - `ReconfigVM_Task` — Generally, you use this operation if you want to reconfigure more properties than just whether or not the virtual machine is a template. One of its parameters, `spec`, is a `VirtualMachineConfigSpec` object. This object contains a boolean

property called `templateOnly` which, when set to true, reconfigures the virtual machine as a template.

- `MarkAsTemplate` — Use this operation if the only part of the configuration you want to change is whether or not the virtual machine is a template. This operation takes as its only parameter the reference to the specific virtual machine that you want to change.

Note: You can change a template back to a virtual machine with the `MarkAsVirtualMachine` operation.

Creating a Virtual Machine from a Template

In SDK 1.x, the template became a virtual machine when you deployed the template. In SDK 2.0, you use an operation to designate the template as a virtual machine.

Creating from a Template in SDK 1.x

You used the `CloneVM` operation to deploy the template as a virtual machine. See [Cloning and Deploying Virtual Machines on page 40](#) for information about using this operation. When you invoked `CloneVM` on a template, the original template was retained and a virtual machine was created from the template.

Creating from a Template in SDK 2.0

As described in [Creating a Template in SDK 2.0 on page 41](#), the template is no longer a separate entity. When you want to create a virtual machine from a template, you invoke the `MarkAsVirtualMachine` operation to change the template to a virtual machine. This clears the boolean property that defines the virtual machine as a template and reassociates the virtual machine with a resource pool and host.

Because the template is a kind of virtual machine, the template no longer exists if you invoke `MarkAsVirtualMachine` on the template. Therefore, if you want to retain the template, you should clone the template with `CloneVM_Task` before invoke `MarkAsVirtualMachine` on the new virtual machine.

Creating a Virtual Disk

In SDK 1.x, you could create a virtual disk using a separate, `CreateVirtualDisk` operation. In SDK 2.0, you no longer create the virtual disk with a separate operation. Instead, you create the virtual disk as part of the `CreateVM_Task` or `ReconfigVM_Task` operation.

Creating a Virtual Disk in SDK 1.x

The `CreateVirtualDisk` operation took two input parameters:

- `vm` — The handle specifying the target virtual machine.
- `diskInfo` — This specifies the parameters that controls the creation of a virtual disk.

Creating a Virtual Disk in SDK 2.0

When you create (`CreateVM_Task`) or reconfigure (`ReconfigVM_Task`) a virtual machine, each operation takes a `VirtualMachineConfigSpec` object as one of its parameters. This object includes an array object, `VirtualDeviceConfigSpec[]` which enables you to define (through the `VirtualDevice` object) the information about a new virtual disk.

Provisioning a Virtual Machine

Provisioning is accomplished in both SDK 1.x and SDK 2.0 using the configuration specification that you provide with the creation operation.

Provisioning in SDK 1.x

As described in [Creating a Virtual Machine from Scratch on page 38](#), you used the Create operation to create a virtual machine in SDK 1.x. One of the parameters passed in to the Create operation was the VirtualMachineSpec. This parameter provided such information as the host for the virtual machine, the guest operating system, and the location of the virtual machine configuration file. Another property of the specification, hardware (of data type VirtualHardware), described the provisioning information for the virtual machine. Through objects such as the VirtualCPUInfo and VirtualMemoryInfo, this property contained information about the CPU, memory, devices, CD-ROMs and so on.

The following sample code snippet shows provisioning in SDK 1.x:

```
public static VirtualHardware createHardware(
    GuestOSInfo guestOS,
    Host host)
    throws Exception {
    VirtualHardware hardware = new VirtualHardware();

    // The VDisk is added as part of the CreateVDisk operation on this VM.
    VirtualCPUInfo CPUInfo = createCPU(guestOS);
    VirtualMemoryInfo memInfo = createMemory(guestOS);
    VirtualNetworkInfo netInfo = createNetInfo();
    VirtualDiskInfo floppyInfo = createFloppy();
    VirtualDiskInfo cdInfo = createCDROM();

    ConfigLimits cfgLimits = host.getInfo().getConfigLimits();
    // Maximum number of network adapters allowed:
    int maxAdapters = cfgLimits.getMaxNet();
    int maxFloppys = cfgLimits.getMaxFloppy();
    // other limits can be accessed similarly

    hardware.setCpu(CPUInfo);
    hardware.setMemory(memInfo);
    hardware.setNet(netInfo);
    hardware.setFloppy(new VirtualDiskInfo[] { floppyInfo });
    hardware.setCd(new VirtualDiskInfo[] { cdInfo });
    return hardware;
}
```

Provisioning in SDK 2.0

The `VirtualMachineSpec` from SDK 1.x maps to the `VirtualMachineConfigSpec` in SDK 2.0. In SDK 2.0, however, you can set the CPU, memory, and device information directly with the properties belonging to the `VirtualMachineConfigSpec` object.

The following sample code snippet shows provisioning in SDK 2.0. The `_vmUtils` object is a user-defined object. The sample also includes a number of user-defined helper functions (`getVmName`, `getAnnotation`, and so on). For the complete code, see `/SDK/samples_2.0/Axis/java/com/vmware/vimsample`.

```
VirtualMachineConfigSpec vmConfigSpec =
    _vmUtils.createVmConfigSpec(getVmName(), null, compresmor, hostmor);

// user specified VM information
vmConfigSpec.setName(getVmName());
vmConfigSpec.setAnnotation(getAnnotation());
vmConfigSpec.setMemoryMB(new Long(getMemorySizeMB()));
vmConfigSpec.setNumCPUs(new Integer(getCpuCount()));
vmConfigSpec.setGuestId(getGuestOsId());

ManagedObjectReference taskmor =
    vimService.createVM_Task(
        vmFolderMor, vmConfigSpec, resourcePool, hostmor
    );
```

Updating Configurations

In SDK 1.x, you invoked the PutUpdates operations when you wanted to make changes to a virtual machine. This operation no longer exists in SDK 2.0. In SDK 2.0, you invoke one of several operations.

Updating Configurations in SDK 1.x

The PutUpdates operation took as its only argument a ChangeReqList data type. This data type comprised an array of Change types (that described the changes) and a handle (for the virtual machine to which the changes applied).

The following steps and sample code snippet use the PutUpdate operation to update the memory setting for a virtual machine.

1. Create the change object.

```
String target = "hardware/memory/sizeMb";
Integer newSize = getMemorySize();
Change change = new Change();
change.setOp(ChangeOp.edit);
change.setTarget(target);
change.setVal(newSize);
Change [] changes = new Change[] { change };
```

2. By using the ResolvePath and GetContents operations, obtain the handle for the virtual machine to which the change is being applied.

```
String handle = serviceConnection.resolvePath(viewPath);
ViewContents viewContents = serviceConnection.getContents(handle);
```

3. Call the PutUpdates operation with this change.

```
ChangeReqList changeList = new ChangeReqList();
ChangeReq changeReq = new ChangeReq();

// For un-versioned putUpdates, send the un-versioned handle
changeReq.setHandle(handle);

// For versioned putUpdates set the versioned handle here:
// changeReq.setHandle(viewContents.getVHandle());
changeReq.setChange(changes);
ChangeReq[] changeReqs = new ChangeReq[] { changeReq };
changeList.setReq(changeReqs);
UpdateList updateList = serviceConnection.putUpdates(changeList);
```

Updating Configurations in SDK 2.0

In SDK 2.0, the PutUpdates operation is replaced with several operations to make changes to objects in inventory:

- ReconfigVM_Task — For changes to a virtual machine object.
- ReconfigureCluster_Task — For changes to a ClusterComputeResource managed object.
- ReconfigureAutostart — For changes to a HostAutoStartManager managed object (the auto-start and auto-stop configuration on a host).
- Reconfigure — For changes to a Task managed object.

Each of these operations takes at least two arguments: a reference to a managed object and a configuration specification object. The latter object describes the changes you want to make to the managed object.

All changes are incremental except for the ReconfigureCluster_Task operation, which takes an additional boolean argument, `modify`. If set to `true`, then the flag specifies that the specification is applied incrementally. If the flag is set to `false` and the operation succeeds, then the configuration of the cluster matches the specification exactly.

The following sample code shows a sample code snippet for the ReconfigVM_Task operation:

```
ManagedObjectReference taskmor = vimService.ReconfigVM_Task(
    vmMor, vmConfigSpec
);
```

The `vmMor` parameter is a reference to the virtual machine managed object being reconfigured. You obtain this managed object reference as described in [Getting Contents Using SDK 2.0 on page 18](#). The `vmConfigSpec` parameter is a `VirtualMachineConfigSpec` object that defines the changes being made.

Connecting and Disconnecting Hosts

In both SDK 1.x and SDK 2.0, you have the ability to connect, reconnect, or disconnect a host from VirtualCenter. The operations that perform this differ between the two releases.

Connecting the Host to VirtualCenter

In SDK 1.x, you connected the host to the VirtualCenter automatically when you added the host. In SDK 2.0, you can set a boolean property that determines whether or not the host is connected. In both SDK 1.x and 2.0, if the host is disconnected, then you can use an operation to reconnect to the VirtualCenter.

Connecting the Host in SDK 1.x

When a host was created (using the Create operation), and a user name and password were supplied during the host creation, then the host was automatically connected to the VirtualCenter and enabled for virtual machine operations.

When the host was in the Disabled state, you used the EnableHost operation to reconnect to the VirtualCenter. The EnableHost operation took one mandatory argument: the handle to the host to be enabled. There were two optional arguments: the user name and password that VirtualCenter used to connect to the host specified by the handle. Upon success, an empty response message was returned.

The following shows a sample code snippet that enabled a host:

```
String handle = serviceConnection.resolvePath("/host/myhost.mydomain.com");
String userName = getUsername(); // optional parameter, null if not supplied
String password = getPassword(); // optional parameter, null if not supplied
serviceConnection.enableHost(handle, userName, password);
```

Connecting the Host in SDK 2.0

In SDK 2.0, you can add a host in either a connected or a disconnected state. You use one of two operations to add the host, depending on whether you want to add a standalone host (AddStandaloneHost_Task) or add a host to a cluster (AddHost_Task). When you add the host, using either of these operations, you determine its state by setting a boolean property, either asConnected (AddHost_Task) or addConnected (AddStandaloneHost_Task). When either of these booleans is set to true, the host is added in the Connected state.

In SDK 2.0, when a host is in the Disconnected state, you use the ReconnectHost_Task operation to reconnect the host to VirtualCenter. The ReconnectHost_Task operation takes one mandatory argument, the managed object reference to the HostSystem that will be reconnected. There is one optional argument, cnxSpec. This HostConnectSpec object contains the parameters to use (including user name and password) when reconnecting to the host. If this parameter is not

specified, the default connection parameters (defined during the AddHost_Task operation) are used.

Disconnecting the Host from VirtualCenter

In SDK 1.x, you used the DisableHost operation. In SDK 2.0, you use the DisconnectHost_Task operation.

Disconnecting the Host in SDK 1.x

You disabled the host from VirtualCenter by using the DisableHost operation. By using the Create operation without supplying the user name and password, you could add a host to the Web service inventory in the Disabled state. The DisableHost operation takes one argument, the handle to the host that will be disabled. Upon success, an empty response message is returned.

```
String handle = serviceConnection.resolvePath("/host/myhost.mydomain.com");
serviceConnection.disableHost(handle);
```

Disconnecting the Host in SDK 2.0

In SDK 2.0, you use the DisconnectHost_Task operation to disconnect a host from the VirtualCenter. When you add a host using either the AddHost_Task or AddStandaloneHost_Task operation, you can add the host in a Disconnected state by setting a boolean property, either asConnected (AddHost_Task) or addConnected (AddStandaloneHost_Task). When either of these booleans is set to false, the host is added in the Disconnected state. You use the ReconnectHost_Task operation (described in [Connecting the Host to VirtualCenter on page 48](#)) to connect the host to VirtualCenter.

The DisconnectHost_Task operation takes one argument, a managed object reference to the HostSystem which is being disconnected. The operation returns a reference to a Task managed object.

Migrating Virtual Machines

In SDK 1.x, there were two operations that moved a virtual machine: `MigrateVM` and `MoveVM`. In SDK 2.0, `MigrateVM` remains (as `MigrateVM_Task`) but `MoveVM` has been replaced with `RelocateVM_Task`. The input parameters have changed for both operations. In addition, a new operation, `ValidateMigration`, enables you to make sure a migration is valid prior to the event.

Hot Migration

In SDK 1.x, `MigrateVM` was the operation you used to migrate virtual machines while they were still powered on. In SDK 2.0, you use `MigrateVM_Task` but the operation has more uses.

Hot Migration in SDK 1.x

In SDK 1.x, you used `MigrateVM` to move a virtual machine in a powered-on state. The parameters were:

- `vm` — The handle for the virtual machine being migrated.
- `host` — The handle to the destination host.
- `priority` — (Optional) Determines whether resources are preallocated before migration starts. Takes one of the following values: `low`, `normal`, or `high`. The default value is “`high`”.
- `dataLocator` — The 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 was ignored in SDK 1.x.

The following sample code snippet shows a hot migration in SDK 1.x:

```
ViewContents task = serviceConnection.migrateVM(
    vmHandle,
    hostHandle,
    priority,
    dataLocator);
```

Hot Migration in SDK 2.0

In SDK 2.0, you use `MigrateVM_Task` to perform a hot migration. In this release, however, you can use `MigrateVM_Task` to migrate a virtual machine in any state: powered-on, powered-off or suspended. If you want to ensure the state of the virtual machine being migrated, you can specify the power state as a parameter. The parameters for `MigrateVM_Task` are:

- `VirtualMachine` managed object reference — The virtual machine to be migrated. The managed object reference is obtained as described in [Getting Contents Using SDK 2.0 on page 18](#).
- `pool` — A reference to a `ResourcePool` managed object. The target resource pool for the virtual machine. If this parameter is left unset, the virtual machine stays assigned to its

existing resource pool, and a host within the same compute resource must be specified. The managed object reference is obtained as described in [Getting Contents Using SDK 2.0 on page 18](#).

- `host` — A reference to a `HostSystem` managed object. The target host to run the virtual machine. This must specify a host that is a member of the `ComputeResource` object indirectly specified by the pool. For a stand-alone host or a cluster with DRS, it can be left unset and the system selects a default from the same `ComputeResource` object as the `ResourcePool` specified by the pool parameter. The managed object reference is obtained as described in [Getting Contents Using SDK 2.0 on page 18](#).
- `priority` — The priority of the migration task. Maps to the `priority` parameter in SDK 1.x.
- `state` — If this parameter is specified, the virtual machine is migrated only if its state matches the specified state.

The `MigrateVM_Task` operation migrates the configuration files, but does not migrate the disk files. See [Moving Files on page 52](#) for more information.

SDK 2.0 and the HostSystem

In SDK 2.0, before you can perform a hot migration, the `HostSystem` to which you are migrating must be properly configured. Every `HostSystem`, through its `configManager` property, gives access to a `vmotionSystem` property. This property is a reference to a `HostVMotionSystem` managed object. This object contains the configuration for a hot migration. For example, the `enabled` boolean determines whether or not a host is enabled for a hot migration. Several operations enable you to configure the `HostVMotionSystem`:

- `Enable/Disable`
- `SelectVNic`
- `UpdateIpConfig`

Cold Migration

In SDK 1.x, you used `MoveVM` to move the disk files, as well as to perform a cold migration of the virtual machine. In SDK 2.0, as described above, you use the `MigrateVM_Task` operation. In addition, you can use a new operation, `RelocateVM_Task`.

Cold Migration in SDK 1.x

In SDK 1.x, you used the `MoveVM` operation to move a virtual machine in the powered-off state. Here are the parameters:

- `vm` — The handle to the virtual machine

- `host` — The 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` — (Optional) The 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.
- `disk` — (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 this parameter is specified, each virtual disk contains two fields:
 - `key` — The key of the virtual disk; for example, `#_scsiDev0:0` or `#ide_Dev1:0`.
 - `dataLocator` — The path describing the destination datastore for the virtual disk file(s).

Cold Migration in SDK 2.0

In SDK 2.0, the `MigrateVM_Task` operation migrates a virtual machine in either a powered-on, powered-off, or suspended state. See [Hot Migration on page 50](#) for an explanation of `MigrateVM_Task` and its parameters. You can also use the `RelocateVM_Task` operation to move a virtual machine in the powered off state. The difference between the two operations is that `MigrateVM_Task` migrates the virtual machine, but does not move the disk files. `RelocateVM_Task` not only migrates the virtual machine, but also moves the disk files. See [Moving Files on page 52](#) for information about the `RelocateVM_Task` operation.

Validating Migration

SDK 2.0 contains a new feature, an operation called `ValidateMigration`. This operation enables you to test a migration before you actually perform the migration using `MigrateVM_Task` or `RelocateVM_Task`.

Moving Files

When you migrate a virtual machine to a different host with the `MigrateVM_Task` operation, only the virtual machine's configuration files are moved. The virtual machine's disk files remain in their original location. To move the disk files, you use the the `RelocateVM_Task` operation. Its parameters are:

- A `VirtualMachine` managed object reference — This is a reference to the virtual machine managed object with the disk files to be moved. You obtain this managed object reference using the process described in [Getting Contents Using SDK 2.0 on page 18](#).
- `spec` — A `VirtualMachineRelocationSpec` object. This object defines the specification of where to relocate the virtual machine. This information includes some or all of the following:

- `datastore` — A reference to a Datastore managed object. The datastore where the virtual machine should be located. If this parameter is not specified, the current datastore is used.
- `disk` - An optional list that allows specifying the datastore location for each virtual disk.
- `host` — The target HostSystem managed object reference for the virtual machine. If not specified, the host association is not changed or is derived from the ResourcePool. You obtain this managed object reference using the process described in [Getting Contents Using SDK 2.0 on page 18](#).
- `pool` — The ResourcePool to which this virtual machine should be attached. For an import operation, the argument is required. For a migrate or clone operation, if the argument is not supplied, the resource pool of the source virtual machine is used. You obtain this managed object reference using the process described in [Getting Contents Using SDK 2.0 on page 18](#).

When you invoke the `RelocateVM_Task` operation to move files, the virtual machine must be powered off. You can also use the `RelocateVM_Task` operation to migrate a powered-off virtual machine to a new host and move its disk files at the same time.

Deleting Hosts and Virtual Machines

See [Deleting Inventory on page 35](#) for porting information about deleting hosts and virtual machines.

4

CHAPTER

Power Operations

This chapter describes how you can modify your client applications between SDK 1.x and SDK 2.0 for power operations related to virtual machines. This includes the following:

- [Powering On a Virtual Machine on page 56](#)
- [Powering Off a Virtual Machine on page 57](#)
- [Suspending a Virtual Machine on page 59](#)
- [Resetting a Virtual Machine on page 60](#)

Powering On a Virtual Machine

When you power on a virtual machine, you start a virtual machine that is either stopped or suspended.

Powering On in SDK 1.x

In SDK 1.x, you used the `ResolvePath` operation to find the handle of the virtual machine you wanted to power on. This operation took as its argument the path (`vmPath`) identifying the virtual machine. Once you found the virtual machine's handle, you passed in this information as an argument to the `StartVM` operation to power on the virtual machine. The following sample code shows this being done.

```
task = serviceConnection.startVM(handle);
```

Powering On in SDK 2.0

In SDK 2.0, the process is similar. Instead of the `ResolvePath` operation, however, you use a `PropertyFilterSpec` to find the virtual machine you want (see [Getting Contents Using SDK 2.0 on page 18](#)). The filter returns a managed object reference to the virtual machine. The `PowerOnVM_Task` operation takes this managed object reference as one of its arguments. The other argument is null and refers to the `HostSystem` managed object.

The following sample code snippet shows how to power on a virtual machine in SDK 2.0. The `vimService` object refers to the URL connection to the server defined during login (see [Logging On on page 13](#)).

```
Task task = vimService.powerOnVM_Task(vmRef, null);
```

The `Task` object enables you to monitor the progress of the operation. You can create another property filter to extract property information about the `Task`.

Powering Off a Virtual Machine

When a virtual machine is running, you can either power off or suspend the virtual machine. In SDK 1.x, you could accomplish either operation using a single method, `StopVM`, with two booleans. In SDK 2.0, you perform these actions using one of the following methods, depending on whether you want to power off or suspend the virtual machine: For information about suspending, see [Suspending a Virtual Machine on page 59](#).

Powering Off in SDK 1.x

To power off a virtual machine in SDK 1.x, first you found the handle for the virtual machine by invoking the `ResolvePath` operation, then you used the `StopVM` method with two boolean properties: `soft` and `suspend`. If you set the `suspend` boolean to `false` and the `soft` boolean to `true`, then you are powering off the virtual machine with a soft shutdown. If you set the `suspend` boolean to `true` and the `soft` boolean to `true`, then you are suspending the virtual machine and placing the guest operating system on standby mode.

Powering Off in SDK 2.0

To perform a soft shutdown in SDK 2.0, first you shut down the guest operating system(s), then you power off the virtual machine.

To shut down the guest operating system(s), you invoke the `ShutdownGuest` operation. Then you invoke the `PowerOffVM_Task` operation on the virtual machine. To perform a hard shutdown, you invoke the `PowerOffVM_Task` operation on the virtual machine.

Performing a Soft Shutdown

In a soft shutdown, first you determine if the guest operating system is running. You do this by checking a property called `guestHeartbeatStatus`. If the operating system is running, then the client shuts down the guest operating and powers off the virtual machine.

The following code snippet performs a soft shutdown in SDK 2.0.

```
vmService.shutdownGuest (vmRef) ;
```

The `vmRef` parameter is a reference to a virtual machine managed object obtained as described in [Getting Contents Using SDK 2.0 on page 18](#).

Performing a Hard Shutdown

In a hard shutdown, if the operation passed in from the command line is `stop`, then the client invokes the `PowerOffVM_Task` operation and powers off the virtual machine.

The following code snippet illustrates a hard shutdown:

```
Task task = vimService.powerOffVM_Task(vmRef) ;
```

The `vmRef` parameter is a reference to a virtual machine managed object obtained as described in [Getting Contents Using SDK 2.0 on page 18](#).

Suspending a Virtual Machine

In SDK 1.x, the program used a boolean to suspend operation. In SDK 2.0, the program uses a new method called `SuspendVM_Task`.

Suspending in SDK 1.x

To suspend a virtual machine in SDK 1.x, you found the virtual machine handle (with `ResolvePath`), then you set a boolean to true: `suspend`. Finally you invoked the `StopVM` method with the virtual machine (derived from `ResolvePath`) as the parameter.

Suspending in SDK 2.0

In SDK 2.0, you use a `PropertyFilterSpec` (see [Getting Inventory Contents on page 18](#)) to find the reference to the virtual machine managed object that you want to suspend. Once you have found the managed object reference, you invoke the `SuspendVM_Task` operation with the reference as the argument.

The following code sample illustrates the suspend operation in SDK 2.0:

```
Task task = vimService.suspendVM_Task(vmRef);
```

Resetting a Virtual Machine

When you want to reset a virtual machine in SDK 2.0, the method is the same as the one used for SDK 1.x.

Resetting in SDK 1.x

In SDK 1.x, you first invoked the `ResolvePath` operation to find the virtual machine, then you invoked the `ResetVM` operation to reset the virtual machine.

Resetting in SDK 2.0

In SDK 2.0, you use a `PropertyFilterSpec` (see [Getting Inventory Contents on page 18](#)) to find the virtual machine. This returns a managed object reference for the virtual machine. This managed object reference is passed in as a parameter to the `ResetVM_Task` operation.

The following code illustrates resetting a virtual machine:

```
Task task = vimService.resetVM_Task(vmRef);
```

In both cases, if the current state is `poweredOn`, then this operation first invokes the `PowerOffVM_Task` operation for a hard shutdown. Once the power state is `poweredOff`, then this operation invokes the `PowerOnVM_Task` operation.

Note: Although this operation powers off then powers on, the two operations are atomic with respect to other clients. Other power operations cannot be performed until the reset method completes.

CHAPTER 5

Event Messages

This chapter describes how you can format event messages. This includes the following:

- [Mapping Events in SDK 1.x and SDK 2.0 on page 62](#)
- [Formatting Event Messages on page 68](#)

Mapping Events in SDK 1.x and SDK 2.0

Events in the SDK 2.0 API are represented as classes defined for each event. This section lists the 1.x Event message and its corresponding 2.0 Event class name. The section also includes the additional events contained in SDK 2.0 for virtual machines and hosts. The SDK 2.0 API contains additional events not shown in this list. Refer to *VMware Infrastructure SDK Reference Guide*.

Corresponding Events Between 1.x and 2.0

The following table shows event messages in SDK 1.x and their corresponding event classes in SDK 2.0.

1.x Event Message		2.0 Event Class Name
ESX Server Messages About the Host or Virtual Machine		2.0 Event Class Name
Type	Message	
info/ warning/ error	Host <host name>: <message>	Multiple Host Events not specifically listed in Host Connection Events in the list below. Please refer to Reference guide.
info/ warning/ error	VM <virtual machine name>: <message>	VmMessageEvent
info/ warning/ error	Virtual Machine <virtual machine name>: <message>	VmMessageEvent
Messages About the User		2.0 Event Class Name
Type	Message	
error	Failed to login user <user name>. Reason: Bad username/password	BadUsernameSessionEvent
error	Failed to login user <user name>. Reason: Already authenticated	AlreadyAuthenticatedSessionEvent
error	Failed to login user <user name>. Reason: No access	NoAccessUserEvent
user	User log entry: <message>	
info	User <user name> logged in	UserLoginSessionEvent
info	User <user name> logged out	UserLogoutSessionEvent
info	VMware VirtualCenter <version> started	ServerStartedSessionEvent
info	VMware VirtualCenter <version> stopped	ServerTerminatedSessionEvent
Host Connection Messages		2.0 Event Class Name
Type	Message	
info	Connected to <host name>	HostConnectedEvent
error	Connection failed for <host name>: Account has insufficient privileges	HostCnxFailedNoAccessEvent

1.x Event Message		2.0 Event Class Name
error	Connection failed for <host name>: Already being managed by <server name>	HostCnxFailedAlreadyManagedEvent
error	Connection failed for <host name>: Bad username/password	HostCnxFailedBadUsernameEvent
error	Connection failed for <host name>: Could not connect to host	HostCnxFailedNoConnectionEvent
error	Connection failed for <host name>: Could not resolve hostname	HostCnxFailedNotFoundEvent
error	Connection failed for <host name>: Failed to configure management account	HostCnxFailedAccountFailedEvent
error	Connection failed for <host name>: Failed to install/upgrade vmware-ccagent	HostCnxFailedCcagentUpgradeEvent
error	Connection failed for <host name>: Incompatible version	HostCnxFailedBadVersionEvent
error	Connection failed for <host name>: Network error	HostCnxFailedNetworkErrorEvent
error	Connection failed for <host name>: Not enough CPU licenses	HostCnxFailedNoLicenseEvent
error	Connection failed for <host name>: Server agent is not responding	HostCnxFailedBadCcagentEvent
error	Connection failed for <host name>: Timeout waiting for	HostCnxFailedTimeoutEvent
error	Connection failed for <host name>: Unexpected error connecting	HostCnxFailedEvent
info	Disconnected from <host name>	HostDisconnectedEvent
error	Lost connection to <host name>	HostConnectionLostEvent
error	Reconnection failed for <host name>	HostReconnectFailedEvent
info	Removed host <host name>	HostRemovedEvent
info	Shutdown of host <host name>, reason: <message>	HostShutdownEvent
Virtual Machine Messages		2.0 Event Class Name
Type	Message	
error	A MAC address (<address>) of <virtual machine name> conflicts with Virtual Machine <virtual machine name>	VmMacConflictEvent
error	A static configured MAC address (<address>) of <virtual machine name> conflicts with Virtual Machine <virtual machine name>	VmStaticMacConflictEvent
info	Powered-on Virtual Machine <virtual machine name> being migrated from host <host name> to host <new host>	VmEmigratingEvent
info	Removed <virtual machine name> from <host name>	VmRemovedEvent
info	<virtual machine name> discovered on host <host name>	VmDiscoveredEvent
warning	<virtual machine name> does not exist on host <host name>	VmOrphanedEvent
info	<virtual machine name> on host <host name> is disconnected	VmDisconnectedEvent
info	<virtual machine name> on host <host name> is powered off	VmPoweredOffEvent (for corresponding Guest operation : VmGuestShutdownEvent)
info	<virtual machine name> on host <host name> is powered on	VmPoweredOnEvent

1.x Event Message		2.0 Event Class Name
info	<virtual machine name> on host <host name> is resetting	VmResettingEvent (for corresponding Guest operation : VmGuestRebootEvent)
info	<virtual machine name> on host <host name> is resuming	VmResumingEvent
info	<virtual machine name> on host <host name> is starting	VmStartingEvent
info	<virtual machine name> on host <host name> is stopping	VmStoppingEvent
info	<virtual machine name> on host <host name> is suspended	VmSuspendedEvent (for corresponding Guest operation : VmGuestStandbyEvent)
info	<virtual machine name> on host <host name> is suspending	VmSuspendingEvent
info	Virtual Machine <virtual machine name> being created on host <host name>	VmCreatedEvent
info	Virtual Machine <virtual machine name> on <host name> has invalid name. Renamed from <old name> to <new name>	VmRenamedEvent
Cloning and Migration Messages		2.0 Event Class Name
Type	Message	
error	BIOS ID (<id>) of <virtual machine name> conflicts with that of <virtual machine name>	VmUuidConflictEvent
info	Changed resource allocation for Virtual Machine <virtual machine name>	VmResourceReallocatedEvent
error	Date of this machine has been rolled back. Disconnecting all hosts.	VmDateRolledBackEvent
error	Failed to clone Virtual Machine <virtual machine name> onto host <host name>. Reason: <message>	VmCloneFailedEvent
error	Failed to create virtual disk <disk name> for virtual machine <virtual machine name>:<message>	VmDiskFailedEvent
error	Failed to deploy template <template name> on host <host name>. Reason:<message>	VmDeployFailedEvent
error	Failed to migrate Virtual Machine <virtual machine name> from host <host name> to host <host name>. Reason: <message>	VmFailedMigrateEvent
error	Failed to <power operation> for virtual machine <virtual machine name> on host <host name>: <message>	VmFailedToPowerOffEvent, VmFailedToPowerOnEvent, VmFailedToRebootGuestEvent, VmFailedToResetEvent, VmFailedToShutdownGuestEvent, VmFailedToStandbyGuestEvent, VmFailedToSuspendEvent
error	License of <name> has expired. Disconnecting all hosts. (Disabling VMotion as well.)	LicenseExpiredEvent (Contains a LicenseFeatureInfo with the list of License Features that have expired)
info	Migration of Virtual Machine <virtual machine name> from host <host name> to host <host name> completed	VmMigratedEvent

1.x Event Message		2.0 Event Class Name
warning	Name for Virtual Machine <virtual machine name> updated on host. Renamed from <old name> to <new name>	VmAutoRenameEvent
error	Number of connected hosts has exceeded available host licenses. Disconnecting all hosts.	HostLicenseExpiredEvent
error	Number of VMotion-enabled hosts exceeds number of available licenses. Disabling VMotion on all hosts.	VMotionLicenseExpiredEvent
info	Powered-off Virtual Machine <virtual machine name> migrating from host <hostname> to host <host name>	VmBeingRelocatedEvent
info	Powered-on Virtual Machine <virtual machine name> is migrating off host <hostname>	VmBeingHotMigratedEvent, VmBeingMigratedEvent
info	Template <template name> being deployed on host <host name>	VmBeingDeployedEvent
info	Template <template name> deployed on host <host name>	VmDeployedEvent
info	The Adapter <adapter> for Virtual Machine <virtual machine name> was assigned a new MAC address: <address>	VmMacAssignedEvent
warning	The destination host <host name> does not have access to the same networks as Virtual Machine <virtual machine name>.	VmNoNetworkAccessEvent
warning	The MAC address for adapter <adapter> in Virtual Machine <virtual machine name> was changed from <old> to <new>	VmMacChangedEvent
info	<virtual machine name> on host <host name> cloned from <virtual machine name> on host <host name>	VmClonedEvent
info	<virtual machine name> on host <host name> is being cloned to <virtual machine name> on host <host name>	VmBeingCloneEvent
Task Messages		2.0 Event Class Name
Type	Message	
info	Task <scheduled task name> created on <entity name>	ScheduledTaskCreatedEvent
info	Task <scheduled task name> on <entity name> completed successfully	ScheduledTaskCompletedEvent
error	Task <scheduled task name> on <entity name> failed. Reason: <message>	ScheduledTaskFailedEvent
error	Task <scheduled task name> on <entity name> failed to send notification to <server>. Reason: <message>	ScheduledTaskEmailFailedEvent
info	Task <scheduled task name> on <entity name> reconfigured	ScheduledTaskReconfiguredEvent
info	Task <scheduled task name> on <entity name> sent notification to <server>	ScheduledTaskEmailCompletedEvent
info	Task <scheduled task name> on <entity name> started	ScheduledTaskStartedEvent
info	Task <scheduled task name> removed from <entity name>	ScheduledTaskRemovedEvent
Alarm Messages		2.0 Event Class Name
Type	Message	
info	Alarm <alarm name> created on <entity name>	AlarmCreatedEvent

1.x Event Message		2.0 Event Class Name
error	Alarm <alarm name> on <entity name> failed to run script <script>. Reason: <message>	AlarmScriptFailedEvent
error	Alarm <alarm name> on <entity name> failed to send e-mail to <email addresses>. Reason: <message>	AlarmEmailFailedEvent
error	Alarm <alarm name> on <entity name> failed to send snmp trap to <server>. Reason: <message>	AlarmSnmpFailedEvent
info	Alarm <alarm name> on <entity name> ran script <script>	AlarmScriptCompleteEvent
info	Alarm <alarm name> on <entity name> reconfigured	AlarmReconfiguredEvent
info	Alarm <alarm name> on <entity name> sent e-mail to <email addresses>	AlarmEmailCompletedEvent
info	Alarm <alarm name> on <entity name> sent snmp trap to <server>	AlarmSnmpCompletedEvent
info	Alarm <alarm name> removed from <entity name>	AlarmRemovedEvent
info	<entity name> caused alarm <alarm name> on <entity name> to change from <color> to <color>	AlarmStatusChangedEvent
info	<entity name> caused action <entity name> of alarm <alarm name> on <entity name> to trigger	AlarmActionTriggeredEvent
Storage Message		2.0 Event Class Name
Type	Message	
error	Multiple datastores named <data store name> detected on host <host name>	DatastoreRenamedOnHostEvent

More Virtual Machine Events

In addition to the event classes listed in the table, SDK 2.0 includes the following virtual machine events.

Migration Event Subclasses

MigrationErrorEvent, MigrationHostErrorEvent, MigrationHostWarningEvent, MigrationResourceErrorEvent, MigrationResourceWarningEvent, MigrationWarningEvent

DRS Events

NoMaintenanceModeDrsRecommendationForVM, NotEnoughResourcesToStartVmEvent, Virtual Machine Creation Events, VmBeingCreatedEvent, VmRegisteredEvent, VmConnectedEvent, VmUuidAssignedEvent, VmUuidChangedEvent

Virtual Machine Reconfiguration Events

VmReconfiguredEvent, VmConfigMissingEvent

Vm Upgrade Events

VmUpgradeCompleteEvent, VmUpgradeFailedEvent, VmUpgradingEvent, VmFailedRelayoutEvent, VmFailedRelayoutOnVmfs2DatastoreEvent, VmRelayoutSuccessfulEvent, VmRelayoutUpToDateEvent

Virtual Machine HA Events

VmDasUpdateErrorEvent, VmDasUpdateOkEvent, VmFailoverFailed

Virtual Machine Relocate Events

VmRelocateSpecEvent - Base Event for Relocate And Clone Events, VmRelocatedEvent, VmRelocateFailedEvent

Vm Resource Pool Events

VmResourcePoolMovedEvent

More Host Events

In addition to the event classes listed in the table, SDK 2.0 includes the following host events.

Host User Account Events

AccountCreatedEvent, AccountRemovedEvent, AccountUpdatedEvent, UserAssignedToGroup, UserPasswordChanged, UserUnassignedFromGroup

Host Datastore Events

DatastoreDiscoveredEvent, DatastorePrincipalConfigured, DatastoreRemovedOnHostEvent

Host DRS Events

DrsResourceConfigureFailedEvent, DrsResourceConfigureSyncedEvent

Host Maintenance Mode Events

EnteredMaintenanceModeEvent, EnteringMaintenanceModeEvent, ExitMaintenanceModeEvent

Host Add Events

HostAddedEvent, HostAddFailedEvent

Host HA Events

HostDasDisabledEvent, HostDasDisablingEvent, HostDasEnabledEvent, HostDasEnablingEvent, HostDasErrorEvent, HostDasOkEvent

Host Datastore Events

LocalDatastoreCreatedEvent, NASDatastoreCreatedEvent, VMFSDatastoreCreatedEvent, General Host Events, CanceledHostOperationEvent, TimedOutHostOperationEvent, HostUpgradeFailedEvent, VcAgentUpgradedEvent, VcAgentUpgradeFailedEvent

Formatting Event Messages

In SDK 1.x, you formatted message strings for each event. In SDK 2.0, you can choose a message string as a property of the event or you can format a message string that is appropriate when viewed from a specific context (for example, the datacenter, host, or virtual machine).

Formatting Event Messages in SDK 1.x

In SDK 1.x, the event message format derived from two objects: a declaration ID and an event declaration.

Each Event object contained a `dec` attribute that contained a declaration ID corresponding to the event. The view, `/event/decls`, represented all the event declarations. This view was an `EventDeclList` object, that contained a single array field called `decl`. Each entry in this array was a separate `EventDecl` (event declaration). By calling `GetContents` on `/event/decls`, the client could obtain all the known event declarations in the system. The event declarations were a set of pre-defined event types that do not get updated. Therefore, clients did not need to call `GetUpdates` on the handle for `/event/decls`.

Each event declaration had the following attributes:

- `key` — String that is the ID of this event declaration.
- `kind` — Type of event, that is one of the following: `"alert"`, `"error"`, `"warning"`, `"info"`, or `"user"`.
- `msgFmt` — Array of format strings that describes how the event message is rendered; for example, `"Task%0 created on %1"`.
- `schedule` — (Optional) Handle of the schedule (if any) that caused this event.

The `key` attribute corresponded to the event's declaration ID. By plugging in the appropriate variable information in the `msgFmt`, you produced the message string.

Formatting Event Messages in SDK 2.0

In SDK 2.0, all event objects have properties to connect them to an associated host, virtual machine, datacenter, and compute resource. They also contain a time stamp, an event ID, and a formatted message describing the event. Message strings can be localized. The localized strings used to create events are stored in the `EventManager`'s `description` array.

To format an event message, you can use a pre-defined string or you can format a message string that is appropriate when viewed from a specific context. To use the pre-defined string, you obtain the value of the Event object's `fullFormattedMessage` property. For example, a powering on event has the following value for its `fullFormattedMessage` property:

```
" {vm.name} on {host.name} in {datacenter.name} is powered on"
```

The `name` property is derived from the `vm`, `host`, and `datacenter` properties of the Event object.

To format a string based on a specific context, you use the properties of the `EventDescriptionEventDetail` data object:

- `formatOnComputeResource` – A string that’s appropriate for the context of a specific cluster. For example, a powering on event in this context produces the following string:

```
"{vm.name} on {host.name} is powered on"
```

- `formatOnHost` – A string that’s appropriate in the context of a specific Host. For example, a powering on event in this context produces the following string:

```
"{vm.name} is powered on"
```

- `formatOnVm` – A string that’s appropriate for the context of a specific virtual machine. For example, a powering on event in this context produces the following string:

```
"Virtual machine on {host.name} is powered on"
```

- `formatOnDatacenter` – A string that’s appropriate in the context of a specific Datacenter. For example, a renaming event in this context produces the following string:

```
"Renamed {vm.name} from {oldName} to {newName}"
```

The `oldName` and `newName` properties are properties of the `vmRenamedEvent` data object.

- `fullFormat` – A string that’s appropriate for the root context. This produces the same string as the `fullFormattedMessage` property of the Event object.

See the *VMware Infrastructure SDK Programming Guide* for some sample code illustrating this usage.

6

CHAPTER

Performance Monitoring

This chapter discusses the differences in how performance monitoring is handled between SDK 1.x and SDK 2.0. The chapter includes the following sections:

- [Performance Monitoring in SDK 1.x and 2.0 on page 72](#)
- [Configuring Intervals for Statistics on page 74](#)
- [Querying Statistics on page 76](#)
- [Querying Metadata Information on page 78](#)

Performance Monitoring in SDK 1.x and 2.0

Performance monitoring in SDK 2.0 shares several concepts with SDK 1.x, but also contains some new concepts.

- Where SDK 1.x had the VirtualCenter Perf Collector to collect statistics for all machines over intervals, SDK 2.0 collects statistics over a set of default and customized intervals.
- In SDK 1.x, you created Filtered Perf Collectors that filtered for specific information. They are children of the VirtualCenter Perf Collectors. In SDK 2.0, you use one of several operations (QueryAvailablePerfMetric, QueryPerf, QueryPerfComposite, or QueryPerfProviderSummary) to query for specific information. However, they have no parent-child relationship with the intervals.
- In SDK 2.0, performance counters represent the statistics metadata, while performance metrics represent the actual statistics collected.

Perf Intervals

In SDK 1.x, a VirtualCenter PerfCollector collected statistics over a performance interval. In SDK 2.0, performance statistics are collected at multiple defined intervals. The information is collected according to the following pre-defined intervals:

- 20 second sample interval, retained one hour.
- Five minute sample interval, retained one day.
- One hour sample interval, retained one week.
- Six hour sample interval, retained 30 days.
- One day sample interval, retained one year.

Each interval includes the frequency with which the statistics are collected as well as the length of the time the statistics are kept in the database. For example, daily performance statistics are stored in the database for one year. This means that one metric value per day is stored for up to one year.

You can define your own intervals with the following restrictions:

- The sample interval must be a multiple of the next shorter interval. Five minutes for one day and seven minutes for two days is not valid. Five minutes for one day and ten minutes for two days is valid.
- The length the stats remain in the database must be longer than the previous setting. Five minutes for two days and ten minutes for one day is not valid, but five minutes for two days and ten minutes for one year is valid.

Each interval is identified by its sampling period (in seconds). This is the interval Id. For example, the interval Id for an interval whose sample interval is five minutes is 300.

Interval Id

In SDK 2.0, each interval is identified by its sample interval (in seconds). This is the interval Id. For example, the interval Id for an interval whose sample interval is five minutes is 300.

CounterIds and MetricIds

In SDK 2.0, performance information is divided into performance counters and performance metrics. A performance counter describes information collected, while a performance metric is the actual information collected. A performance counter is identified by a CounterId. For the entities in question (for example, a host or virtual machine), there might be multiple instances of the device for which the same performance counter can be collected. Each instance is a performance metric. For example, CPU Usage is a performance counter for hosts (represented as host.cpu.usage), while Usage collected for CPU #1 for a host is a performance metric (represented as host.cpu.usage@1). A performance metric is identified by MetricId, which includes the Performance Counter Id and the instance number of the device from which the metric is collected. See *PerformanceManager* in the *VMware Infrastructure SDK Reference Guide* for more detailed information about performance counters and performance metrics.

The query operations described in the following sections each take a MetricId as one of its arguments. The *QueryAvailablePerfMetric* operation returns a list of the MetricIds. From this list, you select the MetricId that you want to use.

Starting Time/Ending Time/Maximum Samples

In SDK 2.0, when you invoke the *QueryPerf* operation, you have the option to specify the starting and/or ending time for the query, as well as the maximum number of samples to be returned. The times specified should correspond to the server time. When the starting time is omitted, the returned metrics start from the first available metric in the system. When the ending time is omitted, the returned result includes up to the most recent metric value. The optional *maxSample* argument specifies the maximum number of samples to be returned by the query. If no *startTime* or *endTime* is specified, but if a *maxSample* is specified, the query returns the most recent *maxSample* number of samples.

Configuring Intervals for Statistics

In SDK 2.0, there are default performance intervals and user-defined intervals as described in [Perf Intervals on page 72](#). You can create, change, or delete intervals.

Creating Performance Intervals

The `CreatePerfInterval` operation enables you to create an interval (in seconds) that is used to gather information for all the hosts and virtual machines. The operation takes two parameters: a managed object reference for the `PerformanceManager` and a `PerfInterval` object. The `PerfInterval` object defines three properties:

- `name` — The user-defined name for the interval.
- `length` — An integer that defines (in seconds) the length of time that information is kept in the database.
- `samplingPeriod` — An integer that defines (in seconds) the interval of the sample. For example, a `samplingPeriod` of 300 samples every five minutes (60x5).

Updating Performance Intervals

You can use the `UpdatePerfInterval` operation to change existing intervals. This operation takes the same parameters as the `CreatePerfInterval` operation. In this case, however, the `samplingPeriod` represents the unique interval identifier of the interval. The `length` and `name` properties are optional and contain the values to be changed.

For example, suppose you have a perf interval of five minutes for one hour. You want to change the length from one hour to two hours. You invoke the `UpdatePerfInterval` operation with the following parameters:

```
PerfInterval interval = new PerfInterval();
interval.samplingPeriod = 300;
interval.length = 7200;
UpdatePerfInterval(pmMor, interval);
```

The first parameter is the managed object reference for the `PerformanceManager`. The second parameter refers to the `PerfInterval` object created prior to invoking to operation.

Note: You cannot change the sampling period for an interval. To change the sampling period, you must first delete the interval, then re-create the interval with a different sampling period.

Remove Performance Intervals

The `RemovePerfInterval` enables you to delete an interval. This operation takes two arguments: the managed object reference to the `PerformanceManager` and the `samplingPeriod`. The number of

seconds in the `samplingPeriod` provides the Interval ID and identifies the specific interval to be removed. For example:

```
RemovePerfInterval(pmMor, 300);
```

Querying Statistics

In SDK 2.0, The following operations enable you to gather statistics:

- QueryAvailablePerfMetric
- QueryPerf
- QueryPerfComposite
- QueryPerfProviderSummary

Retrieving MetricIds

The QueryAvailablePerfMetric operation gets the available performance statistic metrics for a specified ManagedEntity between the optional "beginTime" and "endTime". Those are the performance statistics that are available for querying for the given time interval.

```
queryAvailablePerfMetric(pmMor, meMor, 0900, 1200, 6000);
```

Since the other performance monitoring operations require one or more MetricIds, you would invoke this operation first to get a list of the MetricIds for a certain window in time.

Querying Statistics

The QueryPerf operation retrieves performance statistics for the specified metrics represented by one or more metricIds for all the entities that are listed in the entity argument. The returned statistics are all the available statistics between the optional startTime and endTime. If the optional metricId argument is not provided, all metrics collected for the entity are returned. You must specify at least one entity argument to this query. See [Starting Time/Ending Time/Maximum Samples on page 73](#) for information about using the optional startTime, endTime, and maxSample.

If you omit the optional Interval identifier, then the operation will summarize across all the intervals. For example, suppose you have two intervals, a five minute sample for one hour, and a one hour sample for two days. The database persists 12 samples for each metric from now back to one hour ago, then 23 samples for each metric from one hour ago to 24 hours ago.

Querying Information for an Entity and Its Children

The QueryPerfComposite operation enables you to query for performance statistics for an entity and the breakdown for its child entities. The client can limit the returned information by specifying a list of metrics, and a suggested sample interval ID. The server accepts either the refreshRate property or one of the historical intervals as input interval. For example, you can use this operation to retrieve statistics for a host and all of its registered virtual machines for the given time period. This operation only does one level breakdown. For a resource pool, it only breaks down to its direct child resource pools and virtual machines.

Querying Performance Provider Information

The `QueryPerfProviderSummary` operation retrieves information about a performance provider. This data object type describes capabilities of a performance provider. A performance provider is a `ManagedEntity` that can supply real-time and/or historical statistics. The summary indicates if real-time (current) and historical (summarized) stats are supported. For providers with the `currentSupported` property set to true, clients can call the `QueryPerf` operation with the interval set to the provider's `refreshRate` to retrieve real-time stats. If the `summarySupported` operation is true, clients can call `queryStats` with the interval set to one of the historical intervals configured in the system to retrieve historical statistics for the entity.

Querying Metadata Information

As described earlier in [CounterIds and MetricIds on page 73](#), performance counter information is identified by a CounterId. The QueryPerfCounter operation enables you to retrieve Counter information by passing in one or more CounterIds. For example, suppose you want to find all the counter information related to the group CPU. You can define a helper function which finds all the CounterIds related to CPU and then returns an array of these CounterIds. You use this array of CounterIds as an argument to the QueryPerfCounter operation. The QueryPerfCounter operation also requires a reference to a PerformanceManager managed object:

```
ManagedObjectReference sir = new ManagedObjectReference();
sir.setType("ServiceInstance");
sir.set_value("ServiceInstance");

ServiceContent sic = vimService.retrieveServiceContent(sir);
ManagedObjectReference pmMor = sic.getPerformanceManager();

vimService.queryPerfCounter(pmMor, counterid[]);
```

Glossary

Access Control List

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

Alarm

An entity that monitors one or more properties of a virtual machine, such as CPU load. Alarms use green, red, and yellow color coding to issue notifications as directed by the configurable alarm definition.

Allocated disk

A type of virtual disk in which all disk space for the virtual machine is allocated at the time the disk is created. This is the default type of virtual disk created by VirtualCenter.

Apache Axis

Apache Axis is a more modular, more flexible, and higher-performing SOAP implementation designed around a streaming model and is a successor to Apache SOAP 2.0.

API

Application programming interface. A specified set of functions that allows one to access a module or service programmatically.

Authorization Role

A set of privileges grouped for convenient identification under names such as "Administrator".

Child

A managed entity grouped by a Folder object or other managed entity.

Clone

The process of making a copy of a virtual machine. This process includes the option to customize the guest operating system of the new virtual machine. When a clone is created, VirtualCenter provides an option to customize the guest operating system of that virtual machine. Clones can be stored on any host within the same farm as the original virtual machine.

Cluster compute resource

An extended ComputeResource that represents a cluster of hosts available for backing virtual machines.

ComputeResource

A managed object that represents either a standalone host or a cluster of hosts available for backing virtual machines.

Configuration

See Virtual machine configuration file.

Customization

The process of customizing a guest operating system in a virtual machine as it is being deployed from a template or cloned from another existing virtual machine. Customization options include changing the new virtual machine identification and network information.

Datacenter

An entity used to group virtualmachine and host entities.

Data object

A composite object that is passed by value between the client and the Web service. A data object has properties associated with it but does not have any operations of its own. See also [ManagedObject](#) on page 82.

Datastore

The storage location for the virtual machine files. This may be a physical disk, a RAID, a SAN, or a partition on any of these.

Event

An action that is of interest to VirtualCenter. Each event triggers an event message. Event messages are archived in the VirtualCenter database.

Farm

A structure (in VirtualCenter 1) under which hosts and their associated virtual machines are added to the VirtualCenter server.

Fault

A data object containing information about an exceptional condition encountered by an operation.

Folder

A managed entity used to group other managed entities. The contents of a group are child entities with respect to the Folder object. See [Child](#) on page 80.

A Folder can contain different child entities depending on its location in the ManagedEntity inventory. The `childType` property of the Folder determines the managed entities that can be grouped within the Folder object at any particular level. See the Folder managed object in the *VMware Infrastructure SDK Reference Guide* for more information.

Group

A group is a set of users and groups.

Guest operating system

An operating system that runs inside a virtual machine.

Host

The physical computer on which the virtual machines managed by VirtualCenter are installed.

Host agent

Installed on a virtual machine host, it performs actions on behalf of a remote client.

IDL

Interface definition language. A human-readable syntax used to specify an API. The IDL may be compiled into stubs on a client machine. See [Stub](#) on page 84.

Inventory

A hierarchical structure used by the VirtualCenter server or the host agent to organize managed entities.

JAX-RPC

JAX-RPC (or Java API for XML-based RPC) is an API that builds Web services and clients that use remote procedure calls (RPC) and XML. Remote procedure calls and responses are transmitted as SOAP messages (XML files) over HTTP (the Web).

ManagedEntity

A managed object that is present in the inventory. See [Inventory](#) on page 81.

ManagedObject

A composite object that resides on a server and is passed between the client and the Web service only by reference. A managed object has operations associated with it, but might or might not have properties. See also [Data object](#) on page 80.

Message

A message is a data element that is used by an operation to carry data. It lists the data types exchanged between the Web service and the client.

Microsoft SOAP Toolkit

The Microsoft Simple Object Access Protocol (SOAP) Toolkit 2.0 comprises a client-side component, a server-side component and other components that construct, transmit, read, and process SOAP messages. This toolkit also provides additional tools that simplify application development.

Migration

Moving a virtual machine between hosts. Unless VMotion is used, the virtual machine must be powered off when you migrate it. See [VMotion](#) on page 86.

Nonpersistent mode

If you configure a virtual disk as an independent disk in nonpersistent mode, all disk writes issued by software running inside a virtual machine with a disk in nonpersistent mode appear to be written to disk but are in fact discarded after the virtual machine is powered off. As a result, a virtual disk or physical disk in independent-nonpersistent mode is not modified by activity in the virtual machine.

See also [Persistent mode](#) on page 82.

Operation

A function performed for a client by the Web service.

Permission

A data object comprising an authorization role, a user or group name, and a managed entity reference. Allows a specified user to access the entity with any of the privileges pertaining to the role. See also [Authorization Role](#) on page 80.

Persistent mode

If you configure a virtual disk as an independent disk in persistent mode, all disk writes issued by software running inside a virtual machine are immediately and permanently written to the virtual

disk in persistent mode. As a result, a virtual disk or physical disk in independent-persistent mode behaves like a conventional disk drive on a physical computer.

See also [Nonpersistent mode](#) on page 82.

Privilege

Authorization to perform a specific action or set of actions on a managed entity or group of managed entities. Privileges are grouped together under an authorization role. See also [Authorization Role](#) on page 80.

Property

An attribute of a managed object or data object. A property may be a nested data object or a managed object reference.

PropertyCollector

A managed object used to control the reporting of managed object properties. The primary means of monitoring status on host machines.

Resource pool

A division of computing resources used to manage allocations between virtual machines. Represented by the ResourcePool managed object.

Resume

Return a virtual machine to operation from its suspended state. When you resume a suspended virtual machine, all applications are in the same state they were when the virtual machine was suspended.

See also [Suspend](#) on page 84.

Role

See [Authorization Role](#) on page 80.

Scheduled task

A VirtualCenter activity that is configured to occur at designated times. Represented by the ScheduledTask managed object.

SDK

Software developer kit. It comprises some or all of: an API, an IDL, client stubs, sample code, and manuals.

Service console

The command line interface for an ESX Server system. The service console lets administrators configure the ESX Server system. You can open the service console directly on an ESX Server

system. If the ESX Server system's configuration allows Telnet or SSH connections, you can also connect remotely to the service console.

Service instance

The managed entity at the root of the inventory. Clients must access the service instance to begin a session.

Snapshot

A snapshot preserves the virtual machine just as it was when you took the snapshot — the state of the data on all the virtual machine's disks and whether the virtual machine was powered on, powered off, or suspended. SDK lets you take a snapshot of a virtual machine at any time and revert to that snapshot at any time. You can take a snapshot when a virtual machine is powered on, powered off, or suspended.

SOAP

Simple Object Access Protocol (SOAP) is an XML-based communication protocol and encoding format for inter-application communication in a decentralized, distributed environment. It specifies a standard way to encode parameters and return values in XML, and standard ways to pass them over common network protocols like HTTP (Web) and SMTP (email). SOAP provides an open methodology for application-to-application communication (Web services).

SOAP::LITE

SOAP::Lite is an open source collection of Perl modules that provides a simple and lightweight interface to SOAP.

Stub

A procedure that implements the client side of a remote procedure call. The client calls the stub to perform a task, and the stub then transmits parameters over the network to the server and returns the results to the client.

Suspend

Save the current state of a running virtual machine. To return a suspended virtual machine to operation, use the resume feature.

See also Resume.

To return a suspended virtual machine to operation, use the resume feature.s

Task

A managed object representing the state of a long-running operation.

TCP

Transmission Control Protocol. A reliable transfer protocol between two endpoints on a network. TCP is built on top of IP.

Template

A master image of a virtual machine. This typically includes a specified operating system and a configuration that provides virtual counterparts to hardware components. Optionally, a template can include an installed guest operating system and a set of applications. Templates are used by VirtualCenter to create new virtual machines.

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.

VirtualCenter

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

VirtualCenter agent

Installed on each virtual machine host, it coordinates the actions received from the VirtualCenter server.

VirtualCenter database

A persistent storage area for maintaining status of each virtual machine and user managed in the VirtualCenter environment. This is located on the same machine as the VirtualCenter server.

VirtualCenter server

A service that acts as a central administrator for VMware servers connected on a network. This service directs actions upon the virtual machines and the virtual machine hosts. VirtualCenter server is the central working core of VirtualCenter.

Virtual disk

A virtual disk is a file or set of files that appear as a physical disk drive to a guest operating system. These files can be on the host machine or on a remote file system.

Virtual machine

A virtualized x86 PC environment in which a guest operating system and associated application software can run. Multiple virtual machines can operate on the same host system concurrently.

Virtual machine configuration

The specification of what virtual devices (such as disks and memory) are present in a virtual machine and how they are mapped to host files and devices.

Virtual machine configuration file

A file containing a virtual machine configuration. It is created when you create the virtual machine. It is used by SDK to identify and run a specific virtual machine.

VMFS

See [VMware ESX Server file system](#) on page 86.

VMODL

The interface definition language used in the VMware Infrastructure SDK.

VMotion

The feature that lets you move a virtual machine from one ESX Server system to another without interrupting service. VMotion requires licensing on both the source and target hosts. VMotion is activated by the VirtualCenter agent. The VirtualCenter server centrally coordinates all VMotion activities.

VMware DRS

VMware DRS is a feature that intelligently and continuously balances virtual machine workloads across your ESX Server hosts. VMware DRS detects when virtual machine activity saturates an ESX Server host and triggers automated VMotion live migrations, moving running virtual machines to other ESX Server nodes so that all resource commitments are met.

VMware ESX Server file system

A file system that is optimized for storing virtual machines. Also referred to as VMFS.

One VMFS partition is supported per SCSI storage device or SAN. Each version of ESX Server uses a corresponding version of VMFS. For example, VMFS3 was introduced with ESX Server 3.

VMware HA

VMware HA is a feature that detects failed virtual machines and automatically restarts them on alternate ESX Server hosts. VMware HA selects a failover host that can honor the virtual machine's resource allocations so that service level guarantees remain intact.

VMware Infrastructure

A system of hosts, agents, and clients that communicate to deploy and operate virtual machines. The total VMware solution to managing a data center. See [Host](#) on page 81, [Host agent](#) on page 81, [VirtualCenter](#) on page 85.

VMware Tools

A suite of utilities and drivers that enhances the performance and functionality of your guest operating system. Key features of VMware Tools include some or all of the following, depending on your guest operating system: an SVGA driver, a mouse driver, the VMware guest operating system service, the VMware Tools control panel, time synchronization with the host, VMware Tools scripts, and connecting and disconnecting devices while the virtual machine is running.

Web service

A programming interface based on SOAP and WSDL.

WSDL

Web Services Description Language, an XML-based language used to describe a Web service's capabilities and to provide a way for individuals and businesses to access those services electronically.

XML

Extensible Markup Language, a text-based markup language that is especially designed for Web documents.

A

Performance Counters in 1.x and 2.0

In SDK 1.x, the performance counters were defined as attributes of specific datatypes related to the counter group. For example, the In SDK 2.0, a single data object, PerfCounterInfo, contains information for the counters. The QueryPerfCounter operation returns this object as an array consisting of the performance counter information for the counterIds passed in as one of the parameters. The following shows an example of the CPUPerf datatype in SDK 1.x compared with the PerfCounterInfo data object type in SDK 2.0.

SDK 1.x	SDK 2.0
<pre> CPUPerf { pcpu, used, system, wait } </pre>	<pre> PerfCounterInfo.nameInfo.key </pre> <p>The key property is a string that represents the name of the performance counter: usage, system, wait, and so on.</p>

The following table maps the performance counters in SDK 1.x with their counterparts in SDK 2.0. For complete information about all SDK 2.0 performance counters, refer to the information about the PerformanceManager managed object type in the *VMware Infrastructure SDK Reference Guide*.

Counter Group	SDK 1.x attribute	SDK 2.0 key	Description
CPU Performance	CPUPerf.pcpu	<Not supported>	Physical CPU where this virtual CPU last executed.
	CPUPerf2.pcpu		
	CPUPerf.used	usage	Processor time consumed by each CPU, in milliseconds
	CPUPerf2.used		
	CPUPerf.system	system	System time consumed by each CPU, in milliseconds.
	CPUPerf2.system		
	CPUPerf.wait	wait	Wait time for each CPU, in milliseconds.
	CPUPerf2.wait		
	CPUPerf2.ready	ready	Ready time for each CPU, in milliseconds. A virtual CPU is "ready" when it is runnable but not currently scheduled.
	VirtualMachineCPUPerf.extra	extra	CPU time that is extra.
Memory	MemoryPerf.size	granted	Total amount of managed memory, in KB.
	MemoryPerf2.size		
	VirtualMachineMemoryPerf.size		
	VirtualMachineMemoryPerf2.size		
	MemoryPerf.active	active	Actively used memory, in KB.
	MemoryPerf2.active		
	VirtualMachineMemoryPerf.active		
	VirtualMachineMemoryPerf2.active		
	MemoryPerf2.sizeUnreserved	unreserved	Current amount of unreserved memory, in KB.
	MemoryPerf2.swap	swapped	Total amount of swap space, in KB.

Counter Group	SDK 1.x attribute	SDK 2.0 key	Description
Memory (continued)	MemoryPerf2.swapUnreserved	swapunreserved	Current amount of unreserved swap space, in KB.
	MemoryPerf2.shared	shared	Current amount of shared guest operating system memory, in KB.
	VirtualMachineMemoryPerf2.shared		
	MemoryPerf2.sharedCommon	sharedcommon	Current amount of memory, in KB, consumed for (a single copy of) shared pages.
	MemoryPerf2.heap	heap	Total amount of system heap memory, in KB.
	MemoryPerf2.heapFree	heapfree	Current amount of available system heap memory, in KB.
	VirtualMachineMemoryPerf2.swapped	swapped	Amount of memory being swapped.
	VirtualMachineMemory2.swapTarget	swapTarget	Amount of memory that can be swapped.
	VirtualMachineMemory2.swapIn	swapIn	Amount of memory swapped in.
	VirtualMachineMemory2.swapOut	swapOut	Amount of memory swapped out.
	VirtualMachineMemory2.memCtl	vmmemctl	Amount of memory currently used by the virtual machine memory control.
	VirtualMachineMemory2.memCtlTarget	vmmemctltarget	Target amount of memory the host is trying to balloon for the virtual machine.
	VirtualMachineMemory2.overhead	overhead	Amount of memory that is overhead.

Counter Group	SDK 1.x attribute	SDK 2.0 key	Description
Memory (continued)	MemoryPerf2.state	state	<p>Describes the contention for memory and takes one of the following values. The higher the number, then the memory state is more constrained.</p> <ul style="list-style-type: none"> • 0 — high (lot of memory available) • 1 — soft • 2 — hard • 3 — low (memory is very overcommitted)
Network Performance	Network.packetsIn	packetRx	Packets read from network, for each NIC.
	Network.bytesIn	received	Bytes read from network, for each NIC.
	Network.packetsOut	packetTx	Packets written to network, for each NIC.
	Network.bytesOut	transmitted	Bytes written to network, for each NIC.
Disk Performance	DiskPerf.numRd	numberRead	Number of blocks read in an interval.
	DiskPerf.bytesRd	read	Number of bytes read in an interval.
	GenericDevicePerf.bytesRd		
	DiskPerf.numWr	numberWrite	Number of blocks written in an interval.
	DiskPerf.bytesWr	write	Number of bytes written in an interval.
	GenericDevicePerf.bytesWr		

Revision History

The following table lists the revision history for the *VMware Infrastructure SDK Porting Guide*.

Publication Date	Description
June 15, 2006	Initial publication of the VMware Infrastructure SDK Porting Guide
September 22, 2006	No changes.

Index

A

access control list **79**
alarms **79**
All (boolean property) **22**
allocated disk **79**
API **79**
authorization role **80**

C

CheckForUpdates **26**
child (managed entity) **80**
clone **80**
ClusterComputeResource
and clustering **31**
defined **80**
new object in 2.0 **28**
clustering, create **31**
ComputeResource **80**
Container **28**
CounterId **73**
CreateCluster **31**
CreateFilter **27**
createFolder **31**
CreatePerfInterval **74**

D

data object **80**
Datacenter **28**
Datastore **80**
Disk
performance data **92**

E

ending time **73**
events **80**
formatting messages **68**

F

Farm **28**
farm **81**
fault **81**
Folder **28**
folders **81**

G

GetUpdates **26**
guest operating system **81**
Guest operating system memory **91**

H

Heap memory **91**
High Availability (HA) **86**
host agent **81**
hosts **81**

I

IDL **81**
interval Id **73**
inventory
creating inventory **30**
defined **81**
inventory contents, updates **26**

J

JAX-RPC **81**

M

Managed Entity **82**
managed object
object for property search **18**
retrieving properties **18**
starting point for property search **18**
managed objects **82**
MarkAsTemplate **41, 42**
MarkAsVirtualMachine **42**
maximum samples **73**
message data element **82**
messages, event
formatting **68**

- MetricId **73**
- migration **82**
- N**
- nonpersistent mode **82**
- O**
- Obj property **21**
- ObjectContent data object type **24**
- ObjectSpec **18**
- operation **82**
- P**
- partialUpdates **27**
- PerfCollector **72**
- performance counters **73**
- performance data
 - ending time **73**
 - maximum samples **73**
 - starting time **73**
- performance intervals **72**
- performance metrics **73**
- Permission data object **82**
- persistent mode **82**
- PowerOffVM_Task **57**
- PowerOnVM_Task **56**
- Pressure, memory **92**
- privileges **83**
- properties
 - property search **18**
 - retrieving **18**
- property **83**
- property search
 - defining rules **19**
 - managed object **18**
 - PropertyCollector **19**
 - skip **22**
 - starting point **18**
- PropertyCollector **83**
 - and property search **19**
 - and property updates **27**
- PropertyFilterSpec **18**
- PropertySpec **18**
- Q**
- QueryAvailablePerfMetric **76**
- QueryPerf **73, 76**
- QueryPerfComposite **76**
- QueryPerfProviderSummary **77**
- R**
- ReconfigVM_Task **41**
- RemovePerfInterval **74**
- ResetVM (1.x) **60**
- ResetVM_Task (2.0) **60**
- resource pools **83**
- resume **83**
- RetrieveProperties **19**
- roles **83**
- rules
 - property search **19**
- S**
- scheduled tasks **83**
- SDK **83**
- SelectionSpec **19**
- selectSet property **22, 24**
- ServiceInstanceContent **27**
- Shared guest operating system memory **91**
- ShutdownGuest **57**
- skip (boolean property) **22**
- starting point, property search **18**
- starting time **73**
- startVM **56**
- StopVM **57, 59**
- stubs **84**
- SuspendVM_Task **59**
- Swap space **90**
- System heap memory **91**
- T**
- Task managed object **84**
- template
 - marking as virtual machine **42**

TraversalSpec **19**

Type property **22**

U

Unreserved

swap space **91**

Unreserved memory **90**

UpdatePerfInterval **74**

updates, getting **26**

V

Virtual disk

performance data **92**

virtual machine

changing from template **42**

designating as a template **41**

reconfiguring **41**

virtual machines

powering off **57**

powering on **56**

VirtualMachineGroup **28**

VMware High Availability (HA) **86**

VMware Tools **87**

W

WaitForUpdates **26**

Web service **87**

WSDL **87**

