



# **VI API Webinar Series Virtual Machine Reconfiguration**

Henry Robinson

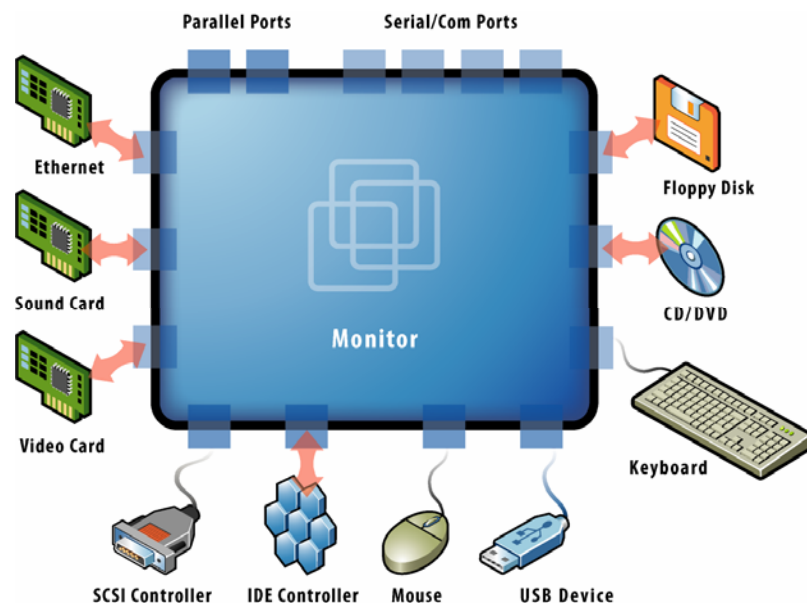
Harvey Alcabes

VMware, Inc.

## Agenda

- ▶ **The Virtual Hardware**
- ▶ **The VI Object Model**
- ▶ **Sample Reconfiguration Examples**
- ▶ **Troubleshooting and Debugging Techniques**

## The Virtual Hardware Architecture



A Virtual Machine consists of a set *processors, memory, and virtual devices*

The VMware Infrastructure allows you to easily modify a Virtual Machine's hardware configuration to respond to an application's needs

VirtualCenter includes a simple-to-use interface to modify the Virtual Hardware Configuration (Edit Settings)

You can also use the VI API to change the Virtual Hardware (ReconfigVM).

# Edit Settings in VirtualCenter

The screenshot displays the VMware Virtual Infrastructure Client interface. The left pane shows a tree view of hosts and clusters, with the 'lamp' virtual machine selected under the host '10.17.213.27'. The main pane shows the 'lamp - Virtual Machine Properties' dialog, which is currently on the 'Hardware' tab. A red arrow points from the 'Edit Settings' button in the bottom toolbar to the 'Hardware' tab. The 'Hardware' tab shows a list of hardware components and their summary information:

Hardware	Summary
Memory	256 MB
CPUs	1
Floppy Drive 1	[/vmfs/volumes/fe2a7...]
CD/DVD Drive 1	[/vmfs/volumes/fe2a7...]
Network Adapter 1	VM Network PEV
SCSI Controller 0	LSI Logic
Hard Disk 1	Virtual Disk

The 'Resources' tab is also visible, showing the memory allocation settings. The memory size is currently set to 256 MB. The 'Memory for this virtual machine:' section includes a slider and a legend with the following values:

- Guest OS recommended minimum: 32 MB
- Recommended memory: 256 MB
- Guest OS recommended maximum: 16384 MB
- Maximum for best performance: 2048 MB

The 'Recent Tasks' pane at the bottom shows a table with columns for Name, Target, Status, and Initiated by.

Name	Target	Status	Initiated by
------	--------	--------	--------------

The VMware logo is visible in the bottom right corner.

## VI API to Modify a Virtual Machine

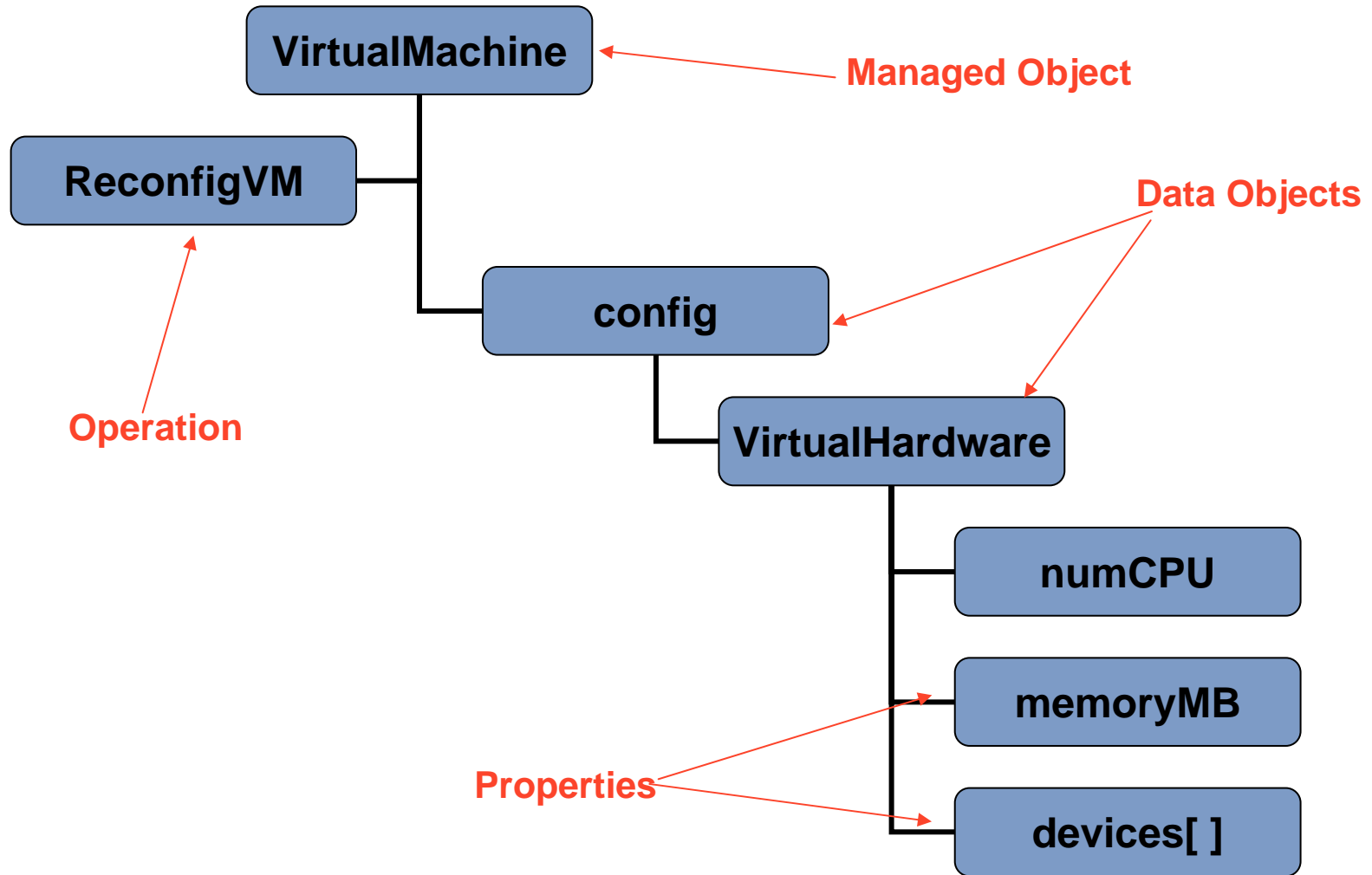
### ► Changing the Virtual Hardware configuration

- During a Clone Operation, you can modify the VirtualMachine configuration specification
  - The new Virtual Machine is deployed with the modified configuration
- You can use the *ReconfigVM* operation to modify an existing Virtual Machine
  - Some operations can only be done with the Virtual Machine powered off: Add memory, add a SCSI controller, etc.
  - Some operations can be done while a Virtual Machine is running: Add a virtual disk, change the Virtual Machine's name, etc.
  - The set of operations that can be done online are dependent on the installed OS.

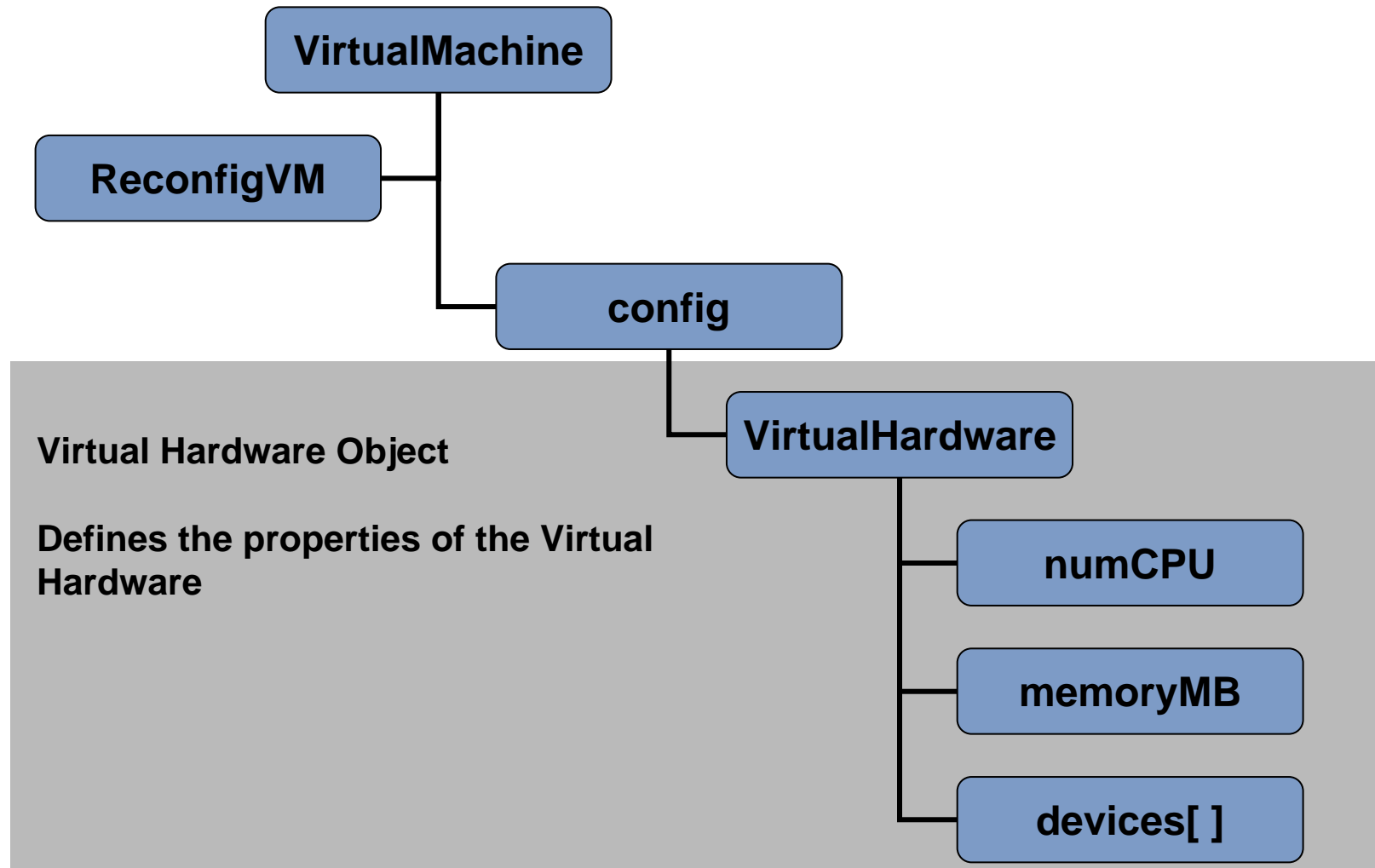
## VI API to Modify a Virtual Machine

1. Retrieve the `VirtualMachine` object for the VM to be changed
2. Build the `VirtualMachineConfigSpec` specification
3. Issue the `ReconfigVM_Task` or `ReconfigVM` (VI Perl Toolkit only)

# The VI Object Model



## The VI Data Model View



## Manipulating the Virtual Hardware

- ▶ **Change the name of the Virtual Machine**
- ▶ **List the Virtual Hardware**
- ▶ **Change the memory size of the Virtual Machine**
- ▶ **Connect a device**
- ▶ **Add a Virtual Disk**
- ▶ **Add a Network Card**
- ▶ **Modify the shares of a Virtual Machine**

## Renaming a Virtual Machine

### exrename.pl (example - Rename)

#### 1. Select the virtual machine.

```
my $vm_name = Opts::get_option ('vm');
my $vm_view = Vim::find_entity_view(view_type => 'VirtualMachine',
                                     filter => {'name' => "^$vm_name\$"});
Fail ("Virtual Machine $vm_name was not found.\n") unless ($vm_view);
```

#### 2. Build the VirtualMachineConfigSpec specification.

```
my $virtualMachineConfigSpec = VirtualMachineConfigSpec->new (
    name => $newName);
```

#### 3. Invoke the ReconfigVM Operation

```
$vm_view->ReconfigVM( spec => $virtualMachineConfigSpec );
```

**Note: Example scripts posted at**

<http://www.vmware.com/vmtn/technology/developer/sample-code/>



# Listing the Virtual Hardware of a Virtual Machine

## exlisthw.pl

### 1. Select the virtual machine.

```
my $vm_name = Opts::get_option ('vm');  
my $vm_view = Vim::find_entity_view(view_type => 'VirtualMachine',  
                                     filter => {'name' => "^$vm_name\$"});  
Fail ("Virtual Machine $vm_name was not found.\n") unless ($vm_view);
```

### 2. Select the *config* data object.

```
my $virtual_hardware = $vm_view->config->hardware;
```

### 3. Retrieve the hardware information

```
print "Information for Virtual Machine:  $vm_name\n";  
printf ("Number of CPUs:                %d\n", $virtual_hardware->numCPU);  
printf ("Memory Capacity (MB): %d\n", $virtual_hardware->memoryMB);
```

***Script also lists the devices from a Virtual Machine.***



## Sample Output

```
C:\VIPERL~1\samples\tests>exlisthw.pl --vm Preetham
Information for Virtual Machine: Preetham
Number of CPUs:      2
Memory Capacity (MB): 384
Summary of Devices
Device              Key  Type
-----
PCI Controller      100 VirtualPCIController
IDE 0                200 VirtualIDEController
IDE 1                201 VirtualIDEController
PS2 Controller      300 VirtualPS2Controller
SIO Controller      400 VirtualSIOController
Video Card           500 VirtualMachineVideoC
Keyboard             600 VirtualKeyboard
Pointing Device     700 VirtualPointingDevic
SCSI Controller     1000 VirtualBusLogicContr
Hard Disk 1         2000 VirtualDisk
Hard Disk 2         2001 VirtualDisk
Hard Disk 3         2002 VirtualDisk
CD/DVD Drive 1     3002 VirtualCdrom
Network Adapter     4000 VirtualPCNet32
Floppy Drive 1     8000 VirtualFloppy
```

# Changing the Virtual Machine Memory Configuration

## exmemory.pl

### 1. Locate the virtual machine.

```
my $vm_name = Opts::get_option ('vm');  
my $vm_view = Vim::find_entity_view(view_type => 'VirtualMachine',  
                                   filter => {'name' => "^$vm_name\$"});  
Fail ("Virtual Machine $vm_name was not found.\n") unless ($vm_view);
```

### 2. Build the Virtual Machine config specification.

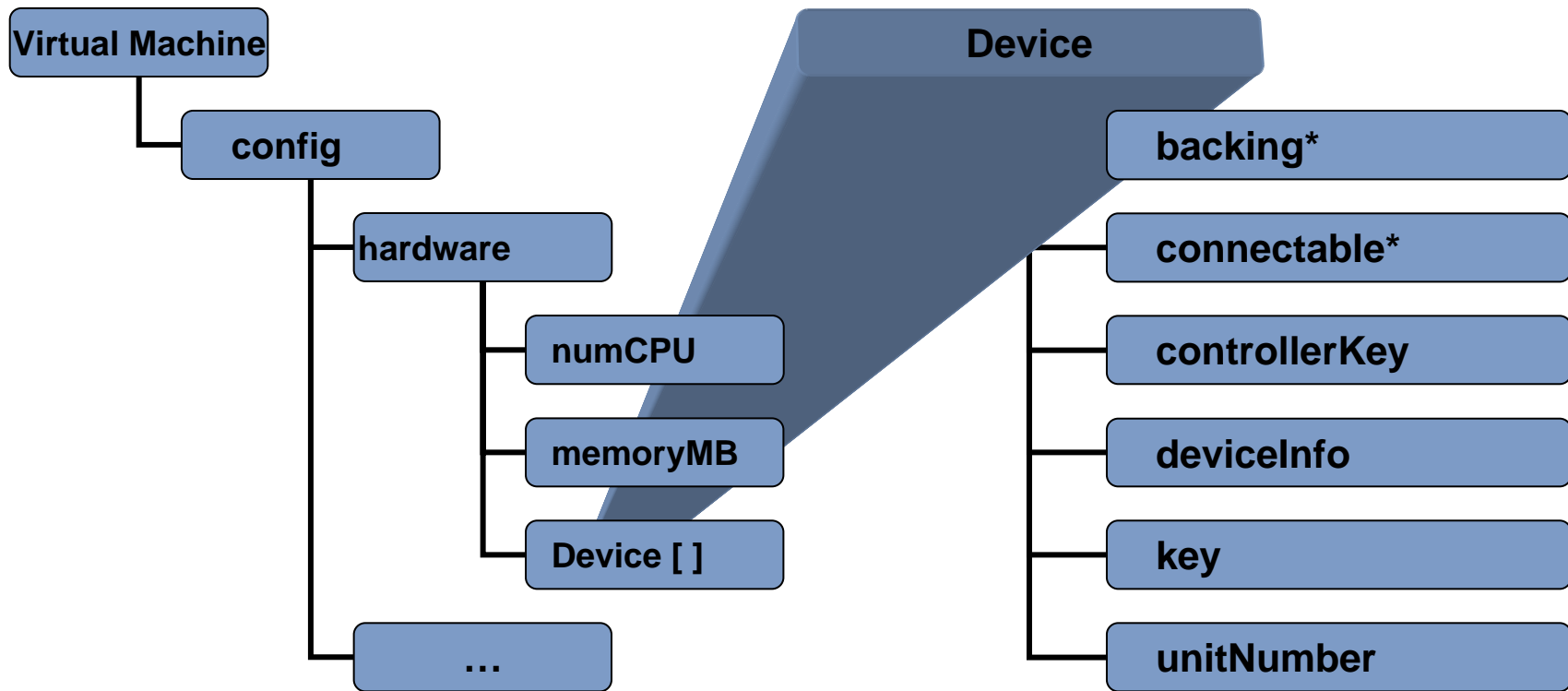
```
my $newSize = Opts::get_option ('memory');  
my $virtual_hardware = $vm_view->config->hardware;  
my $currentMB = $virtual_hardware->memoryMB;  
my $virtualMachineConfigSpec =  
    VirtualMachineConfigSpec->new (memoryMB => $newSize);
```

### 3. Invoke the ReconfigVM Operation

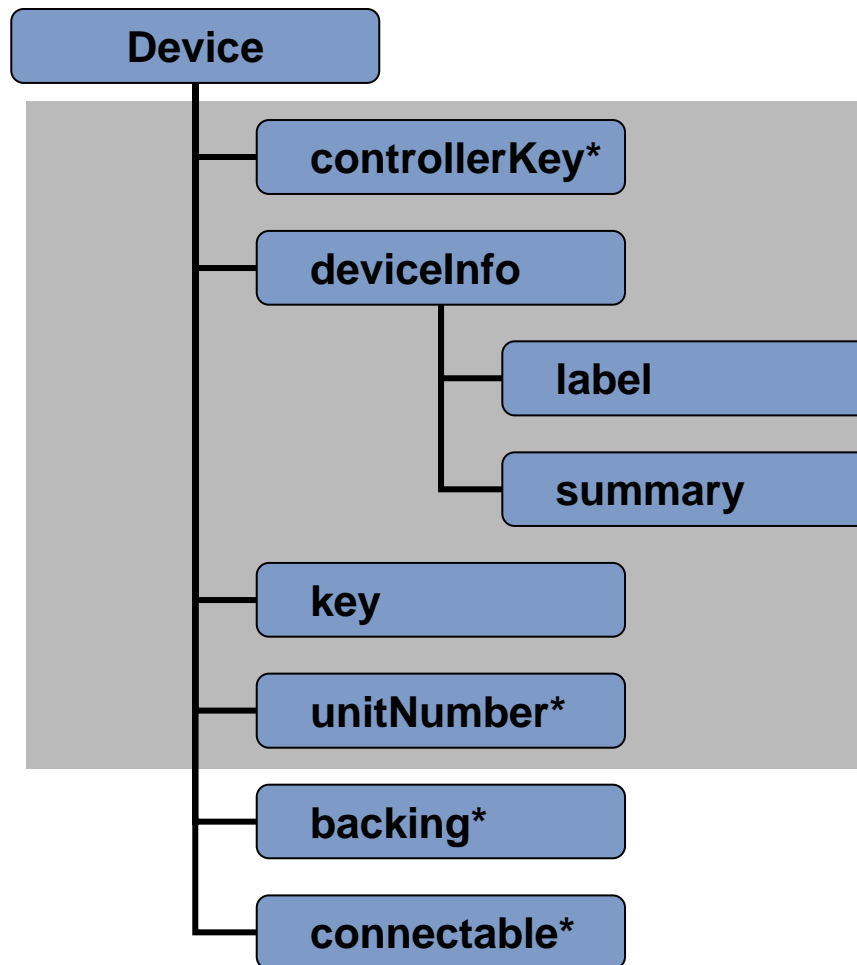
```
$vm_view->ReconfigVM( spec => $virtualMachineConfigSpec );
```

# Virtual Device Object

- ▶ One object per virtual device
- ▶ Stored as an array in the config.virtualHardware



## Virtual Device Object

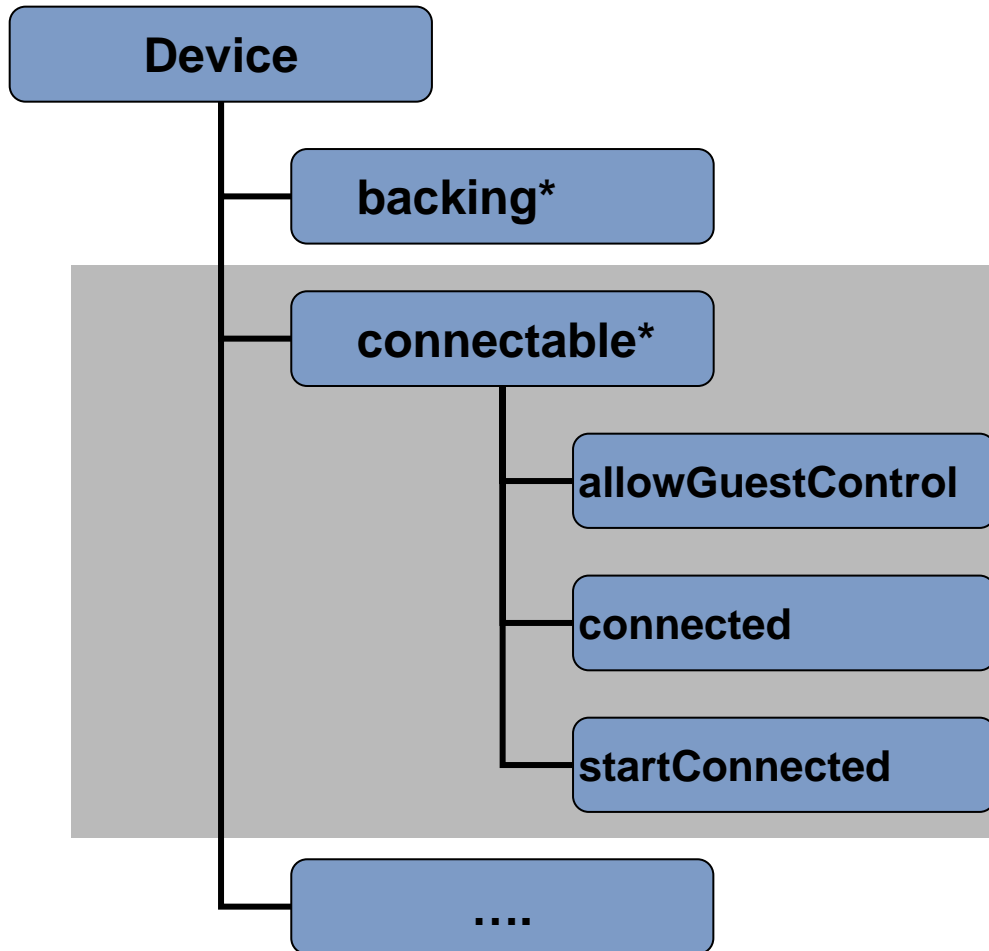


### Basic Properties

- Identifies the device
- The *key* property is unique for each device
- The *label* and *summary* provides a description of the device
- The *controller key* (optional) identifies the key of the controller that controls this device (e.g. SCSI Controller for a disk).

# Virtual Device Object

## The Connectable Object



### The *Connectable* Data Object

Only available for devices that can be “connected”

- Virtual Devices (CD-ROM, floppies)
- Network Cards

## Connecting a CD-ROM

### exconnect.pl

1. **Select the virtual machine.**
2. **Find the CD-ROM Device**

```
my $devName = 'CD/DVD Drive 1';  
my $device = FindDevice ($vm_view, $devName);  
Fail ("Device $devName not found.\n") unless (defined ($device));
```

2. **Verify that the device is currently disconnected.**

```
if ($device->connectable->connected == 0) {
```

3. **Build the spec to connect the device**

```
$device->connectable->connected(1);  
my $devspec = VirtualDeviceConfigSpec->new(  
    device => $device,  
    operation => VirtualDeviceConfigSpecOperation->new('edit');  
my $vmspec = VirtualMachineConfigSpec->new( deviceChange => [$devspec] );
```

4. **Invoke the ReconfigVM Operation**

```
$vm_view->ReconfigVM( spec => $vmspec );
```



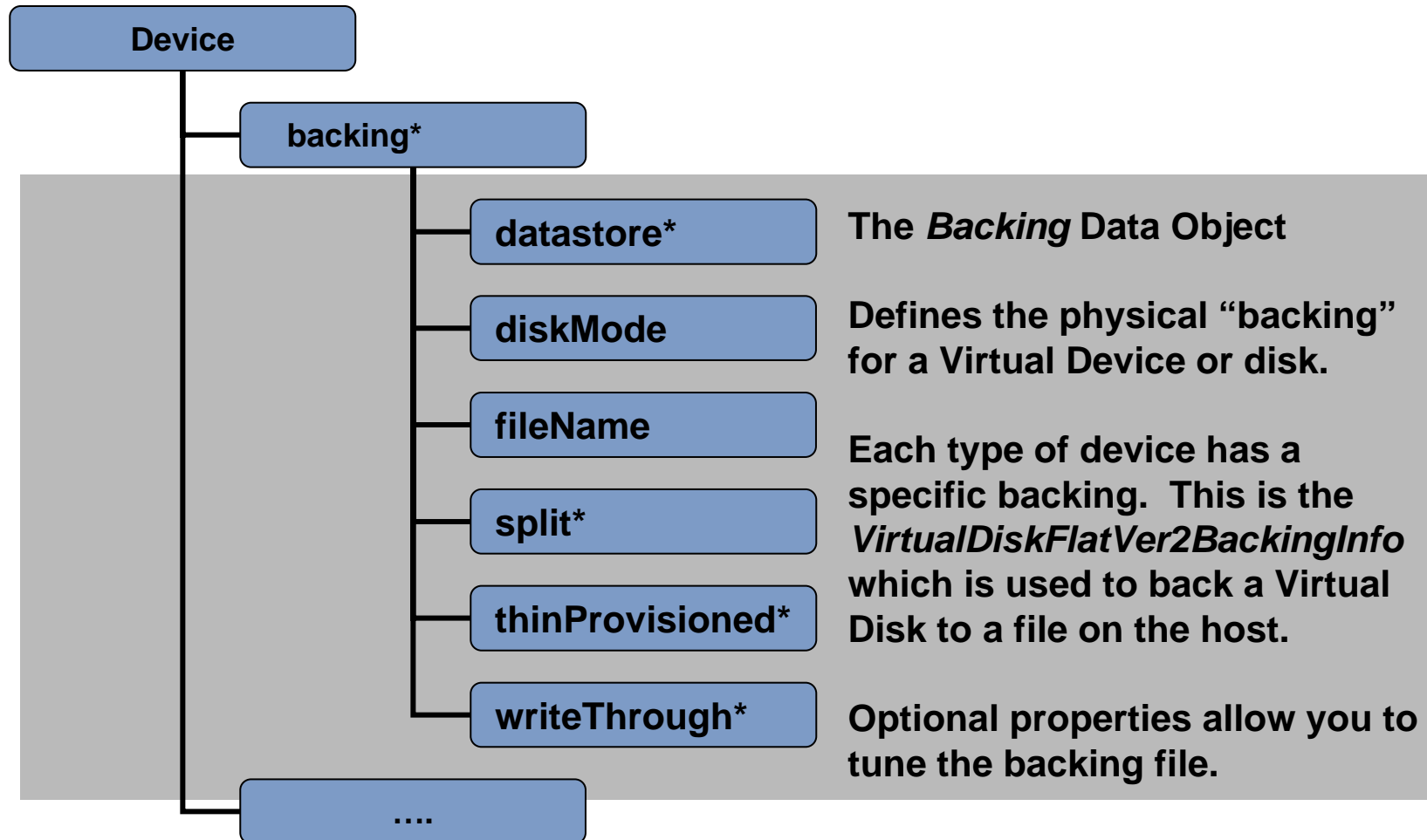
CD/DVD

#### Valid Operations

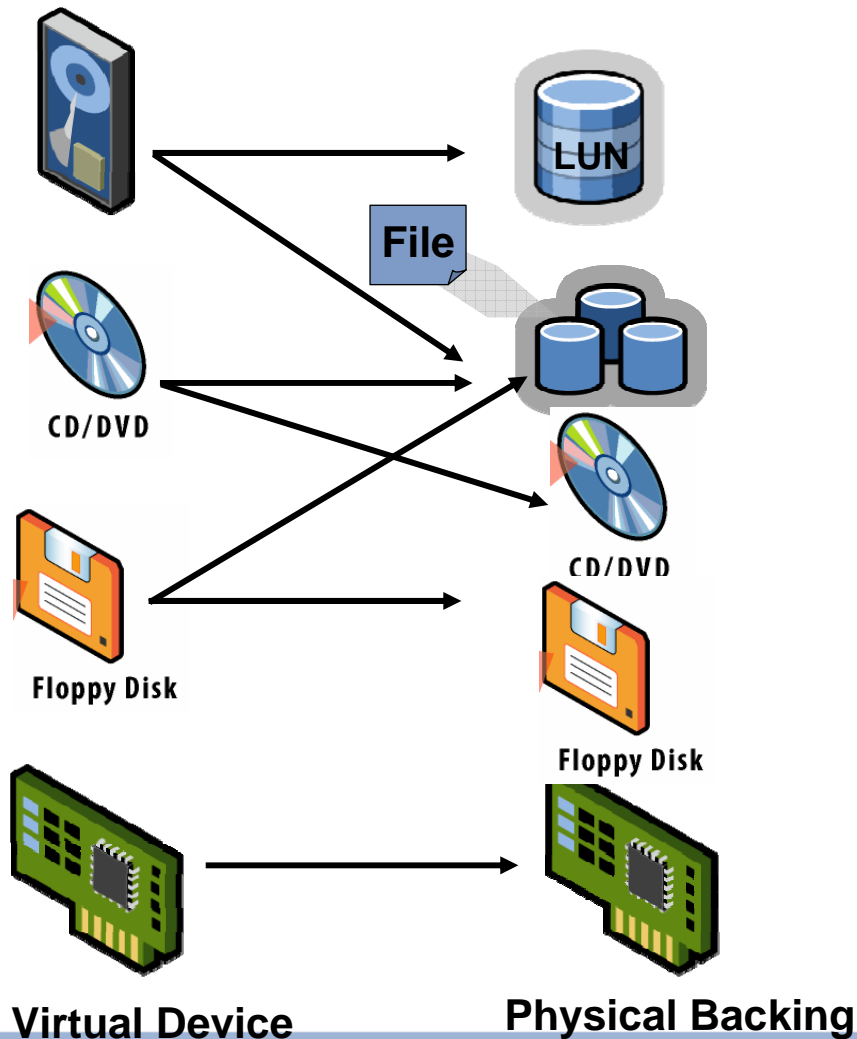
- add
- edit
- remove

# Virtual Device Object

## The Backing Object



## Backing a Virtual Device



### ▶ A VirtualDisk can be backed by:

- A file within the datastore
- A LUN by using raw device mapping

### ▶ A VirtualDevice can be mapped by:

- A physical device
- A ISO file (CD-ROM) or Floppy Image within a datastore

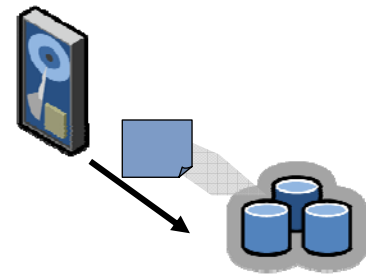
### ▶ A Virtual Network Card can be backed by:

- A Virtual Network connection

## Adding a Virtual Disk

### `exaddvirtualdisk.pl`

1. Build the backing store specification.
2. Build the Virtual Disk specification.
3. **Build the specification to add the device.**
4. **Invoke the ReconfigVM Operation**



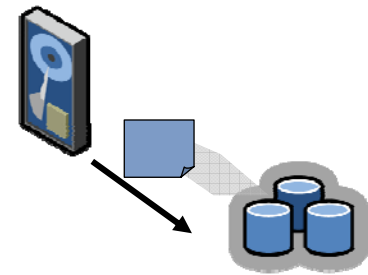
# Adding a Virtual Disk

## Steps to adding a virtual disk

### 1. Build the backing store specification

- ▶ Select the disk mode.

```
my $disk_backing_info =  
    VirtualDiskFlatVer2BackingInfo->new(diskMode => 'persistent');
```



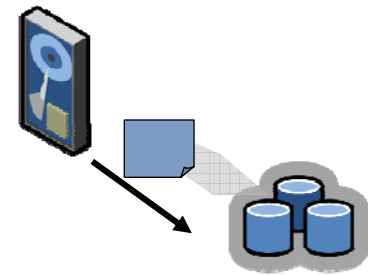
Disk Mode	Snapshot	Changes discarded at VM Reboot
<b>persistent</b>	<b>Yes</b>	<b>No</b>
<b>independent_persistent</b>	<b>No</b>	<b>No</b>
<b>independent_nonpersistent</b>	<b>No</b>	<b>Yes</b>

# Adding a Virtual Disk

## Steps to adding a virtual disk

### 1. Build the backing store specification

- ▶ Select the disk mode.
- ▶ Identify the location to store the Virtual Disk.



```
my $disk_backing_info =  
    VirtualDiskFlatVer2BackingInfo->new(diskMode => 'persistent',  
                                         fileName => '');
```

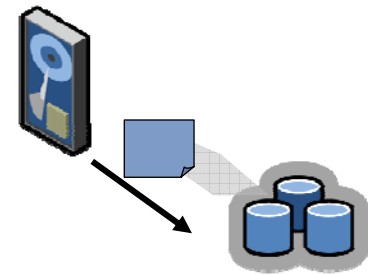
fileName Argument	Effect
Null (i.e. '')	Adds Virtual Disk in the Virtual Machine configuration directory (default name).
[datastore]	Adds Virtual Disk in the Virtual Machine directory for selected datastore.
[datastore] <path> name.vmdk	Adds Virtual Disk in the selected datastore at specified path.

# Adding a Virtual Disk

## Steps to adding a virtual disk

### 1. Build the backing store specification

- ▶ Select the disk mode.
- ▶ Identify the location to store the Virtual Disk.
- ▶ Select the option properties (not required).



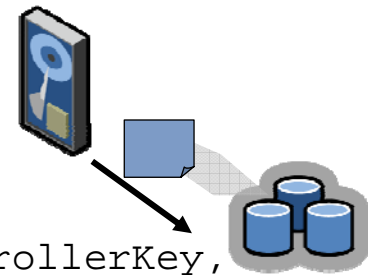
Property	Effect
split*	Splits the Virtual Disk into 2 GB chunks.
thinProvisioned*	Delays allocation until required.
writeThrough*	If set, all write operations are immediately written to disk. Otherwise, simply written to buffer (speed vs reliability).

## Adding a Virtual Disk

1. Build the backing store specification.

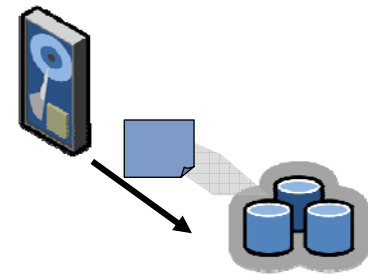
2. Build the Virtual Disk specification.

```
my $disk = VirtualDisk->new( controllerKey => $controllerKey,  
                             unitNumber  => $unitNumber,  
                             key         => -1  
                             backing     => $disk_backing_info,  
                             capacityInKB => $size);
```



## Adding a Virtual Disk

1. Build the backing store specification.
2. Build the Virtual Disk specification.
3. **Build the specification to add the device.**



```
my $operation = VirtualDeviceConfigSpecOperation->new ('add');  
my $fileOperation = VirtualDeviceConfigFileOperation->new ('create');  
my $devspec = VirtualDeviceConfigSpec->new(  
    operation =>$operation,  
    fileOperation => $fileOperation,  
    device => $disk);
```

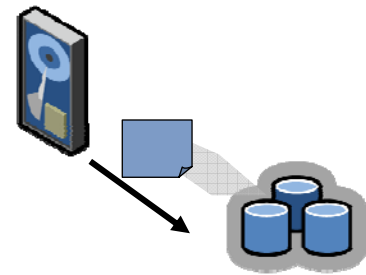
### Valid file operations for “add” operation:

- **create** : create a new file according to the specification
- **<null>** : reuse an existing virtual disk.

## Adding a Virtual Disk

1. Build the backing store specification.
2. Build the Virtual Disk specification.
3. Build the specification to add the device.
4. **Invoke the ReconfigVM Operation**

```
my $vmspec = VirtualMachineConfigSpec->new( deviceChange => [$devspec] );  
$vm_view->ReconfigVM( spec => $vmspec );
```



## Adding a Network Card

### exaddnetwork.pl

### Steps to add a network card

#### 1. Build the backing store specification

```
my $network = "VM Network";  
my $backing_info = VirtualEthernetCardNetworkBackingInfo->new(  
    deviceName => $network);
```

#### 2. Build the network specification

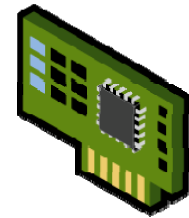
```
my $networkSpec = VirtualPCNet32->new( key => -1,  
    backing => $backing_info);
```

#### 3. Build the reconfig specification

```
my $vmspec = VirtualMachineConfigSpec->new (  
    deviceChange => [$devspec] );
```

#### 4. Execute the ReconfigVM operation

```
$vm_view->ReconfigVM( spec => $vmspec );
```



## Notes about the ReconfigVM Operation

- ▶ **Combine multiple operations into a single call**
  - > For example, you can add multiple devices.
  - > The `VirtualMachineConfigSpec` can contain an arbitrary number of properties to be changed.
- ▶ **Use the *changeVersion* property to protect against simultaneous updates**

```
my $vmspec =  
    VirtualMachineConfigSpec->new (deviceChange => [$devspec],  
                                   changeVersion => $vm_view->config->changeVersion );
```

- ▶ **Wrap your call with *eval* to catch errors**

```
eval {  
    $vm_view->ReconfigVM( spec => $vmspec );  
};  
if ($?) { print "Reconfiguration failed.\n $@"; }  
else {print "Virtual Network card added.\n"; }
```

## How Do I Select Valid Options for Config Specs?

▶ **The QueryConfigOption Operation returns contains information about the execution environment for a virtual machine.**

- List of devices that can be changed online
- Guest operating systems that are supported.
- Capabilities of the virtual machine
- Default devices
- Available backing store types for each device

```
my $env = Vim::get_view (mo_ref => $vm_view->environmentBrowser);  
my $configOption = $env->QueryConfigOption();
```

▶ **See `exlistconfig.pl` for example.**

# Changing the Resource Allocation for a Virtual Machine

## exshares.pl

1. **Select the virtual machine.**
2. **Build the ResourceAllocationInfo object**

```
my $sharesLevel = SharesLevel->new ($levelCpu);  
my $sharesCpu = SharesInfo->new (shares => $cpuShares,  
                                level => $sharesLevel);  
my $resourceAllocationCpu = ResourceAllocationInfo->new (  
                                shares => $sharesCpu);
```

3. **Invoke the ReconfigVM Operation**

```
my $vmspec = VirtualMachineConfigSpec->new (  
                                cpuAllocation => $resourceAllocationCpu);  
$vm_view->ReconfigVM( spec => $vmspec );
```

*For more on shares, see*

[http://pubs.vmware.com/vi301/resmgmt/vc\\_rm\\_%20jumpstart.3.5.html](http://pubs.vmware.com/vi301/resmgmt/vc_rm_%20jumpstart.3.5.html)

## Troubleshooting Techniques

- ▶ Use VirtualCenter as a guide for what can be done (most ReconfigVM features are found under Edit Settings).
- ▶ Refer to the Reference for detailed information on the ReconfigVM Operation (see <http://pubs.vmware.com/vi301/sdk/ReferenceGuide/vi.m.VirtualMachine.html#reconfigure>)
- ▶ Use the MOB to browse through the object structure (<https://<server name>/mob>)

## Summary

- ▶ **ReconfigVM is a powerful operation within the VI API**
- ▶ **With ReconfigVM you can change many aspects of the Virtual Machine environment**