

Troubleshooting Virtual Machine snapshot problems

Ruben Garcia [Mr.Ruben.Garcia@gmail.com]

Last Updated: 02/04/2010

1. [Virtual Machine snapshots](#)
 - [1.1. Understanding Snapshots](#)
 - [1.2. Using the Snapshot Manager](#)
 - [1.3. Snapshots configuration files](#)
 - [1.3.1. The .vmsd and .vmsn files](#)
 - [1.3.2. The .vmdk and -delta.vmdk files](#)
 - [1.3.3. The .vmx file](#)
2. [Troubleshooting snapshots](#)
 - [2.1. Calculating the number of snapshots](#)
 - [2.2. How much space is needed to commit the snapshots of a virtual disk?](#)
 - [2.3. Committing snapshots when there are no snapshot entries in the snapshot manager](#)
 - [2.4. Troubleshooting a failed 'removesnapshots' operation](#)
 - [2.4.1. Corrupt snapshot\(s\) on the chain](#)
 - [2.4.2. Broken CID chain](#)
 - [2.4.3. vmkfstools -q](#)
 - [2.4.4. Hostd behaving abnormally](#)
 - [2.4.5. Check KB](#)
 - [2.5. Other options](#)
 - [2.5.1. Clone the VM](#)
 - [2.5.2. Clone virtual disks](#)
 - [2.5.3. Commit one virtual disk at a time](#)
 - [2.5.4. Commit manually to the base disk](#)
 - [2.5.5. Other options](#)
 - [2.6. How can we free up space on the Datastore?](#)
 - [2.7. General important considerations](#)
 - [2.7.1. Extra considerations](#)
3. [Related KBs](#)
 - [3.1. Public](#)
 - [3.2. Patches](#)
4. [Preventing issues with snapshots](#)
5. [Knowledge gained from experience](#)
6. [Frequently asked questions](#)
 - [6.1. How long is it going to take to delete all the snapshots?](#)
 - [6.2. How do I know when the 'removesnapshots' operation is completed?](#)
 - [6.3. How do I know that the 'removesnapshots' operation is really working?](#)
7. [Useful commands](#)
8. [Acknowledgement](#)
9. [References](#)



Troubleshooting Virtual Machine snapshot problems by Ruben Miguelez Garcia is licensed under a [Creative Commons Attribution-Share Alike 3.0 Unported License](#)

The information in this document is provided "as is" and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information as its sole risk and liability.

This troubleshooting guide explains basic concepts about Virtual Machine snapshots and different troubleshooting paths depending on the problem. This guide was designed for ESX3.5 and [extra considerations](#) have to be taken if working with ESX3.5i or ESX4(i).

The formulas and most of the procedures described in this document were created by the author as part of a continuous troubleshooting improvement process.

1. Virtual Machine snapshots

Virtual Machine (VM) snapshots allow you to preserve the state of the virtual machine so you can return to the same state repeatedly [\[1\]](#).

1.1. Understanding Snapshots

A snapshot captures the entire state of the virtual machine at the time you take it. This includes:

- Settings state : The virtual machine settings (BIOS + .vmx)
- Disk state : The state of all the virtual machine's virtual disks.
- Memory state : The contents of the virtual machine's memory (optional).

When you revert to a snapshot, you return all these items to the state they were in at the time you took that snapshot.

Snapshots are useful when you need to revert repeatedly to the same state but you don't want to create multiple virtual machines or when you are going to apply changes which results you are unsure of.

You must power off the virtual machine before taking a snapshot if the virtual machine has multiple disks in different disk modes. Also Memory cannot be snapshot'ed if any of the disks of the Virtual Machine are in independent mode.

Although you can take an undefined amount of snapshots, VMware supports only up to 32 levels.

When you take a snapshot, be aware of other activity going on in the virtual machine and the likely effect of reverting to that snapshot. In general, it is best to take a snapshot when no applications in the virtual machine are communicating with other computers.

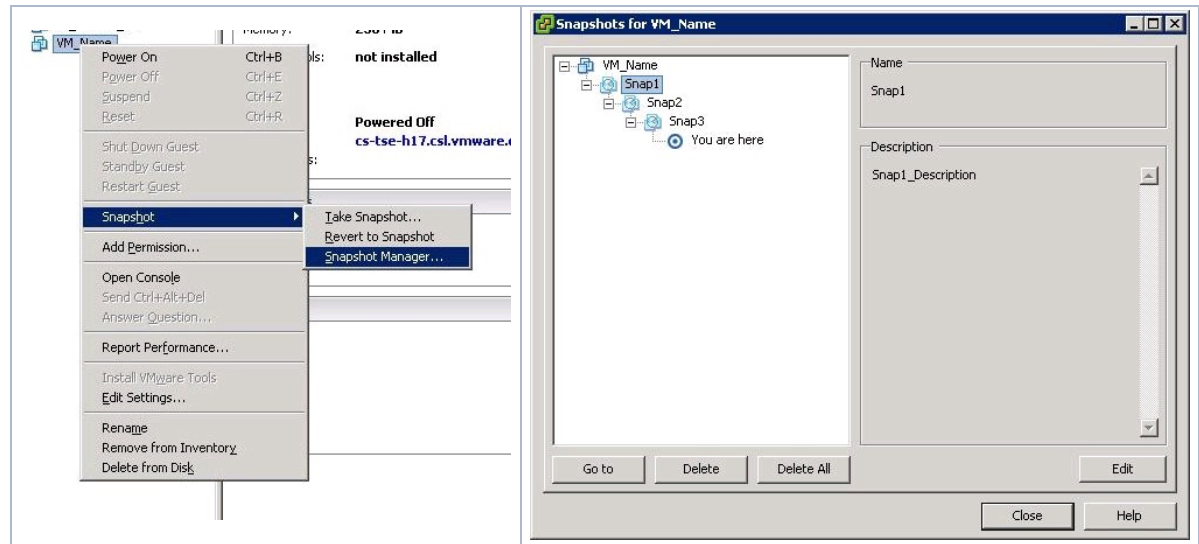
You can take a snapshot while a virtual machine is powered on, powered off, or suspended.

1.2. Using the Snapshot Manager

The Snapshot Manager lets you review all the snapshots for the active virtual machine and act on them directly.

- The **Go to** command allows you to restore the state of any snapshot.
- **Delete** commits the snapshot data to the parent and then removes the selected snapshot.
- **Delete All** commits all the immediate snapshots before the *You are here* current state to the base disk and removes all existing snapshots for that virtual machine.

Throughout all this document we will use delete and commit as synonyms. Do not interpret 'delete' as deleting physically a file. Why do we use delete instead of commit? Because the snapshot manager has only 2 buttons for the operation of committing snapshots and their names are 'Delete' and 'Delete all'.



1.3. Snapshots configuration files

Every time a snapshot is created several files are created or updated. They are the following:

- .vmsd**
Snapshot descriptor file
- .vmsn**
Snapshot settings file
- .vmdk and -delta.vmdk**
Snapshot disk/delta files
- .vmx**
VM configuration file

Each incremental snapshot description is held in the `.vmsd`, the settings and virtual memory (optional) is kept in the `.vmsn` file.

1.3.1. The .vmsd and .vmsn files

This `.vmsd` file contains the structure of the snapshot tree. It describes the characteristics of every snapshot : with or without memory, UID numbers, names of related files, etc.

The reality is that this file gets corrupt very easily and that is when the snapshots problem begins. In this guide we will not explain in detail the characteristics and structure of this file since during the troubleshooting process it is commonly ignored.

See [2] if you are interested in a deeper knowledge of these files.

1.3.2. The .vmdk and -delta.vmdk files

The `.vmdk` is just a descriptor of the `-delta.vmdk`. It contains as well a pointer to the parent `.vmdk`

The `.vmdk` of a snapshot is just like the `.vmdk` of a Base Disk. Below we will use the command `sgrep` to see only the lines with relevant information. The only difference is that the Base Disk has no parent.

```
# sgrep VM_Name-000002.vmdk
CID=7d8e30a3
parentCID=8d4aba54
parentFileNameHint="VM_Name-000001.vmdk"
RW 1638400 VMFSSPARSE "VM_Name-000002-delta.vmdk"
```

Here is the explanation of every line:

CID=7d8e30a3	Content ID: 8 Digit hex number identifying the disk
parentCID=8d4aba54	The parent's Content ID
parentFileNameHint="VM_Name-000001.vmdk"	Path and name of the parent disk
RW 1638400 VMFSSPARSE "VM_Name-000002-delta.vmdk"	Logical size of the snapshot (same than the BaseDisk) and name of the delta file.

As you can see, the `.vmdk` of the Base Disk is pretty similar:

```
# sgrep VM_Name.vmdk
CID=7d8e30a3
parentCID=ffffffff
```

```
RW 1638400 VMFS "VM_Name-flat.vmdk"
```

The `-delta.vmdk` is the file that contains the changes done on this snapshot in relation with its parents. It is like a `-flat.vmdk` with a few differences:

- Its content has no meaning by itself because it contains only changes in reference to the content of its parents. Its content is consistent as far as its parents have not been modified since its creation.
- It can grow, up to the size of its Base Disk.

If for any reason the `.vmdk` disappears, you can recreate it, however the `-delta.vmdk` is irreplaceable.

1.3.3. The `.vmx` file

As you may know this is the VM settings configuration file and it will be in most cases the first place we will check in the troubleshooting process.

We will come back to this file later, so for now let's see only what is interesting for us in the snapshots context.

If you check its content with `sgrep` we may see something like this

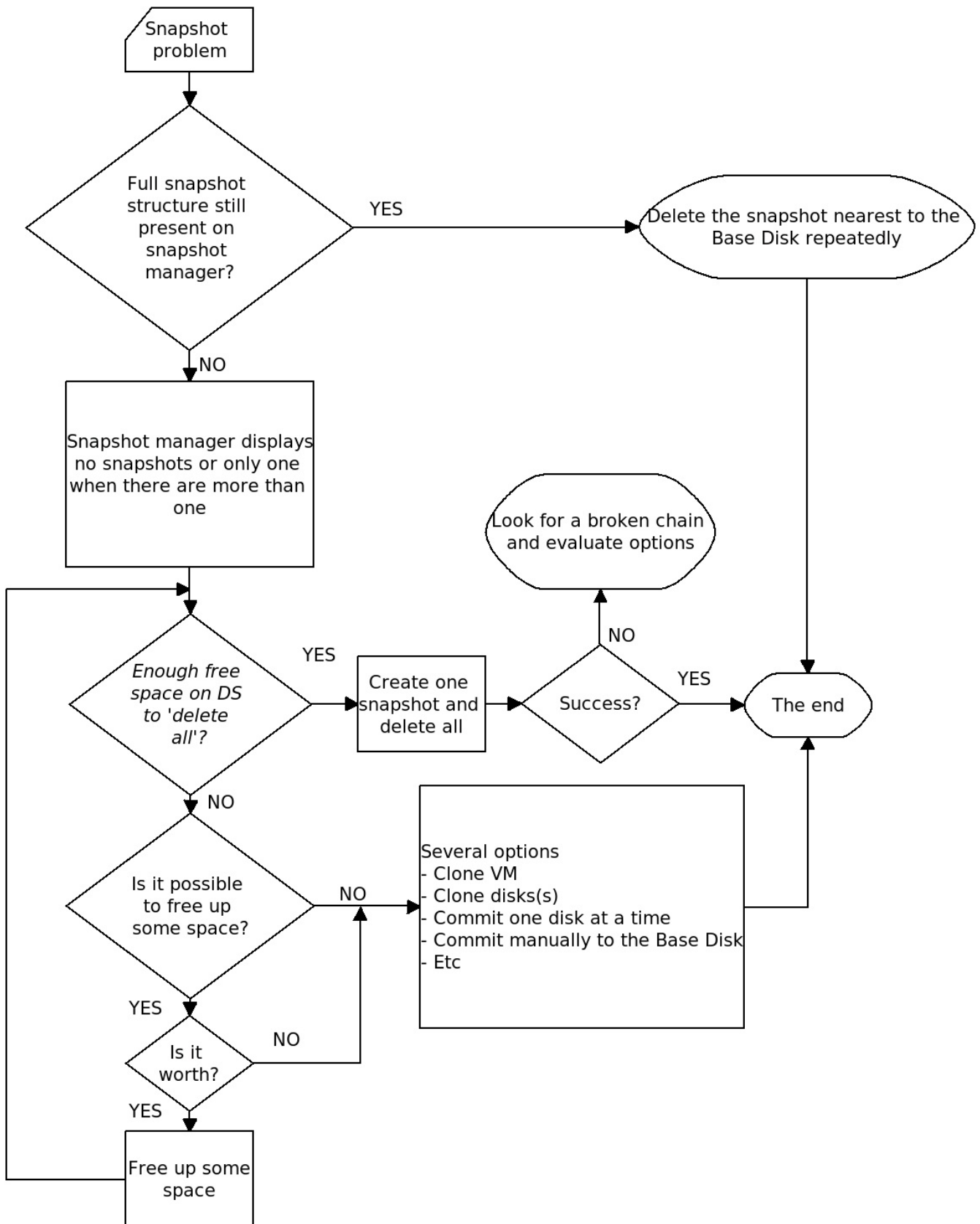
```
# sgrep VM_Name.vmx
scsi0:0.present = "true"
scsi0:0.fileName = "VM_Name.vmdk"
```

<code>scsi0:0.present = "true"</code>	This indicates that the disk in <code>scsi0:0</code> is presently attached to the VM
<code>scsi0:0.fileName = "VM_Name.vmdk"</code>	This is the name of the disk attached on <code>scsi0:0</code>

2. Troubleshooting snapshots

The severity of the '*snapshots problem*' can go from trivial (customer has no knowledge of how snapshots work) to really critical and very time consuming (only one VM with several huge snapshots in the only Datastore which is full).

Use the flowchart below to help you in the troubleshooting process. Many issues have more than one solution, so you may need to evaluate the options and the criticality of the system. Don't be afraid to tell the customer that you need some time to evaluate the options and have no rush checking everything before starting implementing your plan. There is no way to stop or revert a '*removesnapshots*' operation.

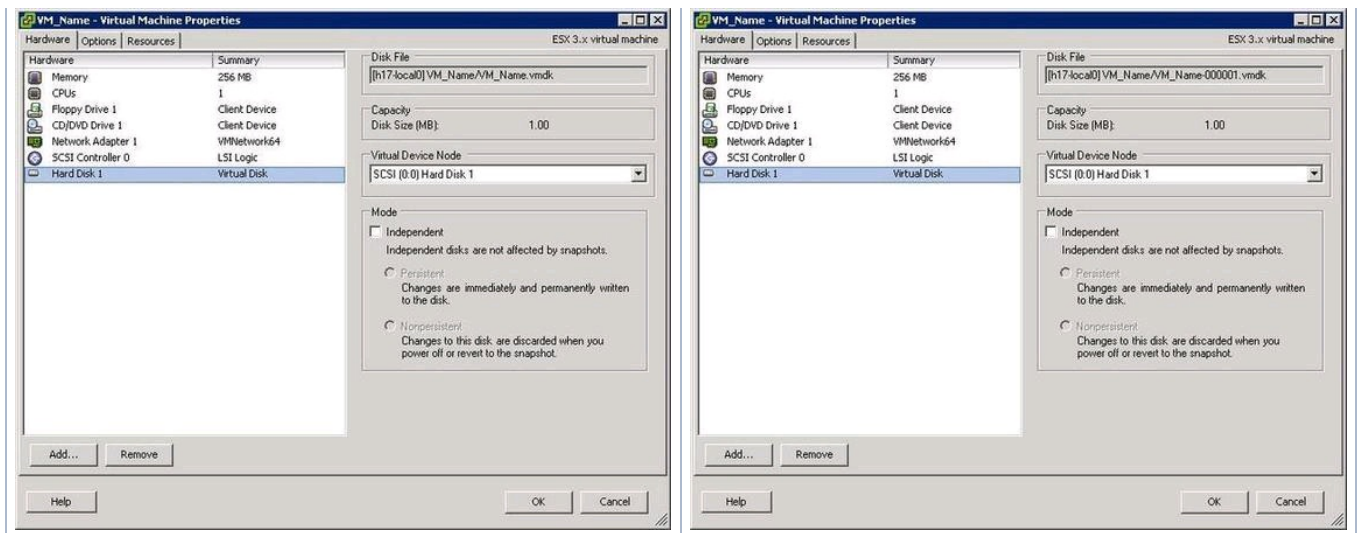


2.1. Calculating the number of snapshots

You can't know how many snapshots a VM has just looking to the dates and names of the files in the VM folder. You need first to check the [.vmtx](#) file and find out how many disks the VM has and if they are using snapshots. You can check this using 'Edit Settings' on the VI Client as well, but that is all you can do on that window, so if the VM has snapshots, then you will have to go to the Service Console to continue troubleshooting.

On the VI Client -> VM Settings , you can see something like this:

VM without snapshots	VM with snapshots
----------------------	-------------------



As you may have noticed, the only difference is the file name.

If we `sgrep` the `.vmx` we will find in both cases:

VM without snapshots	VM with snapshots
# <code>sgrep VM_Name.vmx</code>	# <code>sgrep VM_Name.vmx</code>
<code>scsi0:0.present = "true"</code>	<code>scsi0:0.present = "true"</code>
<code>scsi0:0.fileName = "VM_Name.vmdk"</code>	<code>scsi0:0.fileName = "VM_Name-000001.vmdk"</code>

The result on the left means that this VM has only one virtual disk and it is NOT running on snapshots. No matter what is on the VM folder, right now this VM is not using any snapshot.

But if you have an output like the right one, you know that that virtual disk is running on snapshots.

Here is another example. This VM has 4 virtual disks and the VM is running on snapshots on all of them.

```
# sgrep VM_Name.vmx
scsi0:0.present = "true"
scsi0:0.fileName = "VM_Name-000003.vmdk"
scsi0:1.present = "true"
scsi0:1.fileName = "VM_Name_1-000003.vmdk"
scsi0:2.present = "true"
scsi0:2.fileName = "VM_Name_2-000003.vmdk"
scsi0:3.present = "true"
scsi0:3.fileName = "VM_Name_3-000003.vmdk"
```

The question is, how many snapshots does it have? To know it you need to follow the chain of `.vmdk` files until you get to the Base Disk. And you have to do the same with the other disks attached to the VM if you want to have the complete picture. You can not make assumptions here if you really want to know the truth. The VM may have several disks, virtual or RDMs, and each one can have a different number of snapshots, and the base disk may also be located on a different datastore. The number on the file name means nothing. The numbers on the chain may be sorted or may be not.

```
# sgrep VM_Name-000003.vmdk
CID=7d8e30a3
parentCID=7d8e30a3
parentFileNameHint="VM_Name-000008.vmdk"
RW 1638400 VMFSSPARSE "VM_Name-000003-delta.vmdk"

# sgrep VM_Name-000008.vmdk
CID=7d8e30a3
parentCID=7d8e30a3
parentFileNameHint="VM_Name-000002.vmdk"
RW 1638400 VMFSSPARSE "VM_Name-000008-delta.vmdk"

# sgrep VM_Name-000002.vmdk
CID=7d8e30a3
parentCID=7d8e30a3
parentFileNameHint="VM_Name.vmdk"
RW 1638400 VMFSSPARSE "VM_Name-000002-delta.vmdk"

# sgrep VM_Name.vmdk
CID=7d8e30a3
parentCID=ffffffff
RW 1638400 VMFS "VM_Name-flat.vmdk"
```

You can also `sgrep` them all and find the chain on the results:

```
# sgrep VM_Name*[1-9].vmdk
VM_Name-000001.vmdk:CID=7d8e30a3
VM_Name-000001.vmdk:parentCID=7d8e30a3
VM_Name-000001.vmdk:parentFileNameHint="VM_Name.vmdk"
VM_Name-000001.vmdk:RW 1638400 VMFSSPARSE "VM_Name-000001-delta.vmdk"
VM_Name-000002.vmdk:CID=7d8e30a3
VM_Name-000002.vmdk:parentCID=7d8e30a3
VM_Name-000002.vmdk:parentFileNameHint="VM_Name-000001.vmdk"
VM_Name-000002.vmdk:RW 1638400 VMFSSPARSE "VM_Name-000002-delta.vmdk"
VM_Name-000004.vmdk:CID=7d8e30a3
VM_Name-000004.vmdk:parentCID=7d8e30a3
VM_Name-000004.vmdk:parentFileNameHint="VM_Name.vmdk"
VM_Name-000004.vmdk:RW 1638400 VMFSSPARSE "VM_Name-000004-delta.vmdk"
[...]
```

```

VM_Name_3-000002.vmdk:CID=ade33caf
VM_Name_3-000002.vmdk:parentCID=ade33caf
VM_Name_3-000002.vmdk:parentFileNameHint="VM_Name_3-000001.vmdk"
VM_Name_3-000002.vmdk:RW 8192 VMFSSPARSE "VM_Name_3-000002-delta.vmdk"
VM_Name_3-000004.vmdk:CID=ade33caf
VM_Name_3-000004.vmdk:parentCID=ade33caf
VM_Name_3-000004.vmdk:parentFileNameHint="/vmfs/volumes/47b31f65-74a19d98-e78b-001a4bb24256/VM_Name/VM_Name_3.vmdk"
VM_Name_3-000004.vmdk:RW 8192 VMFSSPARSE "VM_Name_3-000004-delta.vmdk"

```

2.2. How much space is needed to commit the snapshots of a virtual disk?

Imagine the following situation:

```

Base Disk
+ Snap1
+ Snap2
+ ...
+ SnapN
+ You are here

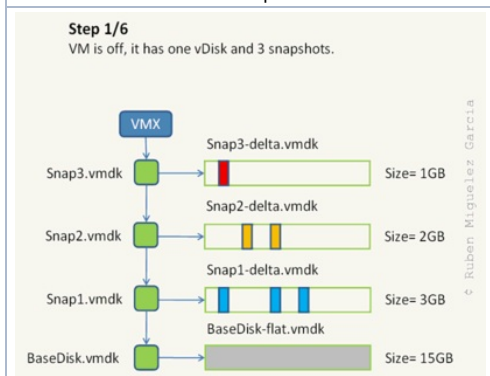
```

When a snapshot is deleted, its content is merged into its parent file. Often a lot of the same disk blocks will change which means the size of Snap_i would not become Snap_i + Snap_{i+1}, but it is better to be conservative and do the calculations for the worst case scenario, otherwise we could end up with a completely full datastore and an incomplete *Delete* task with undetermined data consistency on the snapshot files.

Premises:

- The Base Disk never grows.
- No snapshot is deleted until the whole task is completed (it may be *Delete* on one snapshot or *Delete all*).
- Each snapshot delta file can grow up to the size of its Base Disk.
- Each virtual disk has its own snapshot(s) file(s).
- Committing Snap1 to the Base Disk does not require additional space.
- The VM snapshot files are kept in the VM's directory.
- If the VM is running during snapshot consolidation there will be a temporary delta file created. This temporary file is used to store all the changes made while the snapshots are being consolidated. Therefore its size depends on the I/O activity inside the virtual machine.

Process behind 'Delete all' snapshots.



If the VM is off, the space needed to delete the first N snapshots can be calculated as:

$\text{TotalUsed} = \text{SUM}\{i=1 \text{ to } N\} (\text{MIN}(\text{SUM}\{j=i \text{ to } N\} (S_j) , BD))$	$\text{TotalUsed} = \sum_{i=1}^N \min \left(\sum_{j=i}^N S_j , BD \right)$
$\text{TotalOriginal} = \text{SUM}\{i=1 \text{ to } N\} (S_i)$	$\text{TotalOriginal} = \sum_{i=1}^N S_i$
$\text{TotalNeeded} = \text{TotalUsed} - \text{TotalOriginal}$	$\text{TotalNeeded} = \text{TotalUsed} - \text{TotalOriginal}$

Where

TotalUsed

Summation of the sizes of the snapshots after being inflated with the content of their children.

TotalOriginal

Summation of the sizes of the snapshots before start committing them.

TotalNeeded

Free space needed on the datastore where the snapshots are located to be able to delete them all on the worst case scenario.

BD

Size of Base Disk

S_i

Size of Snapshot_i

N

Number of snapshots. See [Calculating the number of snapshots](#) to know N, and S_i.

However in most cases

$$\text{TotalOriginal} \leq \text{BD} \quad \text{TotalOriginal} \leq \text{BD}$$

So if the condition above is True, then we can calculate the space needed easily with:

$$\text{TotalNeeded} = \text{SUM}(i=1 \text{ to } N) (S_i * (i-1))$$

$$\text{TotalNeeded} = \sum_{i=1}^N S_i(i-1)$$

Important: This calculation has to be done for every disk on the VM. The total amount of space needed is the summation of space needed to commit the snapshots of every disk.

2.3. Committing snapshots when there are no snapshot entries in the snapshot manager

This is the source of this procedure: Committing snapshots when there are no snapshot entries in the snapshot manager (<http://kb.vmware.com/kb/1002310>)

The process is very simple. It only requires the use of `vmware-cmd` command. Do `vmware-cmd -h` for more information about this command or check the documentation.

If you query the VM and it reports to have no snapshots:

```
# vmware-cmd VM_Name.vmx hassnapshot
hassnapshot() =
```

Then you move away the `.vmsd` file to avoid further confusions on the VM:

```
# mv VM_Name.vmsd /tmp/
```

Then you create one snapshot:

```
# vmware-cmd VM_Name.vmx createsnapshot SnapshotName
createsnapshot(SnapshotName) = 1
```

And now that it recognizes that it has snapshots, you can request to delete all with:

```
# vmware-cmd VM_Name.vmx removesnapshots
removesnapshots() = 1
```

If it fails, you will get an error message on the Service Console and something like this on the VI Client. To know why it has failed you will need to [check several things and correct what is wrong](#)

Recent Tasks							
Name	Target	Status	Initiated by	Time	Start Time	Complete Time	
Remove Snapshot	VM_Name	A general system error occurred: Internal error	root	21/01/2009 16:56:02	21/01/2009 16:56:02	21/01/2009 16:56:04	
Update Resource Pool C...	Resources	Completed	vpuser	21/01/2009 16:54:27	21/01/2009 16:54:27	21/01/2009 16:54:27	
Update Resource Pool C...	Resources	Completed	vpuser	21/01/2009 16:54:27	21/01/2009 16:54:27	21/01/2009 16:54:27	

2.4. Troubleshooting a failed 'removesnapshots' operation

Firstly, it is always worth taking a look to the VM logs (`vmware.log`), the vmkernel logs (`/var/log/vmkernel`) and the hostd logs (`/var/log/vmware/hostd.log`) to see if you find there the reason why it failed.

In many cases, even if you see the reason there you will have to fix the problem by yourself. Lets see the most common issues that cause a 'removesnapshots' or a cloning operation to fail:

2.4.1. Corrupt snapshot(s) on the chain

If the size of the delta file is $\leq 16\text{Mb}$ you can consider that it contains no information about the changes on the VM and can be safely ignored if it is not situated between two valid snapshots on the snapshot chain.

On the table below you can see that the minimum size of the snapshot doesn't depend on the size of the Base Disk.

-rw-----	1	root	root	16M	Jan 16 15:36	VM_Name-000001-delta.vmdk
-rw-----	1	root	root	219	Jan 16 15:36	VM_Name-000001.vmdk
-rw-----	1	root	root	1.0M	Jan 15 16:41	VM_Name-flat.vmdk
-rw-----	1	root	root	305	Jan 16 15:33	VM_Name.vmdk
-rw-----	1	root	root	16M	Jan 16 15:38	VM_Name-000001-delta.vmdk
-rw-----	1	root	root	221	Jan 16 15:38	VM_Name-000001.vmdk
-rw-----	1	root	root	100M	Jan 15 16:41	VM_Name-flat.vmdk
-rw-----	1	root	root	309	Jan 16 15:37	VM_Name.vmdk
-rw-----	1	root	root	16M	Jan 16 15:40	VM_Name-000001-delta.vmdk
-rw-----	1	root	root	222	Jan 16 15:40	VM_Name-000001.vmdk
-rw-----	1	root	root	1.0G	Jan 15 16:41	VM_Name-flat.vmdk
-rw-----	1	root	root	311	Jan 16 15:39	VM_Name.vmdk
-rw-----	1	root	root	16M	Jan 16 15:41	VM_Name-000001-delta.vmdk
-rw-----	1	root	root	223	Jan 16 15:41	VM_Name-000001.vmdk
-rw-----	1	root	root	10G	Jan 15 16:41	VM_Name-flat.vmdk
-rw-----	1	root	root	313	Jan 16 15:41	VM_Name.vmdk

-rw-----	1	root	root	16M	Jan 16 15:42	VM_Name-000001-delta.vmdk
-rw-----	1	root	root	223	Jan 16 15:42	VM_Name-000001.vmdk
-rw-----	1	root	root	20G	Jan 15 16:41	VM_Name-flat.vmdk
-rw-----	1	root	root	313	Jan 16 15:42	VM_Name.vmdk

So if the .vmdk is pointing at the SnapN and you see something like this (logical representation of the chain):

```
SnapN (<16MB) -> SnapN-1 (>=16MB) -> .... -> BaseDisk
```

You can safely modify in the .vmx the reference to SnapN for SnapN-1 if you are trying to commit them all or simply point at SnapN-1 if you are creating a clone disk.

2.4.2. Broken CID chain

Every .vmdk file contains his own CID (Content ID) and the CID of its parent. The parent CID of the Base Disk is always 'ffffff'. If a commit snapshots/clone disk operation fails, you need to verify that the chain of CIDs is correct for every virtual disk on the VM.

Usually, if you find on the vmware.log file something like "The base disk or one of the snapshots it depends on has been modified" then there is a broken CIDs chain.

You can check it in many different ways. You can for example sgrep all the .vmdk descriptors and examine the results.

```
# sgrep Tony\ test\ VM-00000[1-9].vmdk
Tony test VM-000001.vmdk:CID=4c652f89
Tony test VM-000001.vmdk:parentCID=0112ccc5
Tony test VM-000001.vmdk:parentFileNameHint="Tony test VM.vmdk"
Tony test VM-000001.vmdk:RW 1048576 VMFSSPARSE "Tony test VM-000001-delta.vmdk"
Tony test VM-000002.vmdk:CID=91889aa4
Tony test VM-000002.vmdk:parentCID=4c652f89
Tony test VM-000002.vmdk:parentFileNameHint="Tony test VM-000001.vmdk"
Tony test VM-000002.vmdk:RW 1048576 VMFSSPARSE "Tony test VM-000002-delta.vmdk"

# sgrep -H Tony\ test\ VM.vmdk
Tony test VM.vmdk:CID=0112ccc5
Tony test VM.vmdk:parentCID=ffffff
Tony test VM.vmdk:RW 1048576 VMFS "Tony test VM-flat.vmdk"
```

Also, if you know the chain in advance, you can see only the CIDs in order doing something like this:

```
# grep CID Tony\ test\ VM-00000{2,1}.vmdk
Tony test VM-000002.vmdk:CID=91889aa4
Tony test VM-000002.vmdk:parentCID=4c652f89
Tony test VM-000001.vmdk:CID=4c652f89
Tony test VM-000001.vmdk:parentCID=0112ccc5

# grep -H CID Tony\ test\ VM.vmdk
Tony test VM.vmdk:CID=0112ccc5
Tony test VM.vmdk:parentCID=ffffff
```

Or all at once with

```
# grep CID Tony\ test\ VM-00000{2,1}.vmdk Tony\ test\ VM.vmdk
Tony test VM-000002.vmdk:CID=91889aa4
Tony test VM-000002.vmdk:parentCID=4c652f89
Tony test VM-000001.vmdk:CID=4c652f89
Tony test VM-000001.vmdk:parentCID=0112ccc5
Tony test VM.vmdk:CID=0112ccc5
Tony test VM.vmdk:parentCID=ffffff
```

As you may have noticed, [Bash wildcards](#) are very useful if you know how to use them.

Warning: If the CIDs chain is broken it is probably because the Base Disk or one of the snapshots behind the last one has been modified after the creation of the last snapshot. In that case you have to explain to the customer that you [can not guaranty consistency](#) of the data if they have been really modified and in that case, it is [better](#) to create a clone of the whole chain and use that clone instead. To be able to create the clone you will have to modify the CIDs chain manually to make it consistent.

If the CID chain has got broken after a failed 'delete all' operation then it is safe (in most cases) to fix the CID chain and commit the snapshots. However, if you want to be **completely** sure that the content inside the VM is fine, power it on and verify its content **before** you commit the snapshots.

2.4.3. vmkfstools -q

This command is useful for something more than just querying RDMs.

```
# vmkfstools -q --queryrdm <file.vmdk>
List the attributes of a raw disk mapping. When used with a
`rdm:<device>` specification, it prints out the
vml of the raw disk corresponding to the mapping
referenced by the <device>. It also prints out identification
information for the raw disk (if any).
```

When you use "vmkfstools -q" against a vmdk file it will walk through all the chain (if it exists) until the Base Disk. It will check that all the vmdk descriptor files are consistent with each other and that there are no missing files. Basically it will check that everything is there and makes sense. However it will not analyze the content inside the flat/delta files, just their existence.

2.4.3.1. Finding CID broken with vmkfstools

Nothing wrong

```
# vmkfstools -q VM_Name2-000002.vmdk
VM_Name2-000002.vmdk is not an rdm
```


CID broken

```
# vmkfstools -q VM_Name2-000002.vmdk
Failed to open 'VM_Name2-000002.vmdk' : The parent virtual disk has been modified
since the child was created (18).
```

With verbose output

```
# vmkfstools -q VM_Name2-000002.vmdk -v 10
DISKLIB-VMFS : "/VM_Name2-000002-delta.vmdk" : open successful (23) size =
83886080, hd = 0. Type 8
DISKLIB-VMFS : "/vmfs/volumes/4b72a71c-0047198b-03f3-
00151725d513/VM_Name2/VM_Name2-000001-delta.vmdk" : open successful (23) size =
83886080, hd = 0. Type 8
DISKLIB-VMFS : "/vmfs/volumes/4b72a71c-0047198b-03f3-
00151725d513/VM_Name2/VM_Name2-flat.vmdk" : open successful (23) size = 83886080, hd
= 0. Type 3
DISKLIB-LINK : Attach: Content ID mismatch (6fef7bec != 86fef7be).
DISKLIB-CHAIN : "/vmfs/volumes/4b72a71c-0047198b-03f3-
00151725d513/VM_Name2/VM_Name2.vmdk" : failed to open (The parent virtual disk has
been modified since the child was created).
DISKLIB-VMFS : "/VM_Name2-000002-delta.vmdk" : closed.
DISKLIB-VMFS : "/vmfs/volumes/4b72a71c-0047198b-03f3-
00151725d513/VM_Name2/VM_Name2-000001-delta.vmdk" : closed.
DISKLIB-VMFS : "/vmfs/volumes/4b72a71c-0047198b-03f3-
00151725d513/VM_Name2/VM_Name2-flat.vmdk" : closed.
DISKLIB-LIB : Failed to open 'VM_Name2-000002.vmdk' with flags 0x17 (The parent
virtual disk has been modified since the child was created).
Failed to open 'VM_Name2-000002.vmdk' : The parent virtual disk has been modified
since the child was created (18).
```

Explanation

DISKLIB-LINK : Attach: Content ID mismatch (6fef7bec != 86fef7be).	The CID chain is broken
DISKLIB-CHAIN : "/vmfs/volumes/4b72a71c-0047198b-03f3-00151725d513/VM_Name2/VM_Name2.vmdk"	Between this file and its child

2.4.3.2. Finding RW mismatch with vmkfstools

RW mismatch

```
# vmkfstools -q VM_Name2-000002.vmdk -v 10
DISKLIB-VMFS : "/VM_Name2-000002-delta.vmdk" : open successful (23) size =
595886080, hd = 0. Type 8
DISKLIB-VMFS : "/vmfs/volumes/4b72a71c-0047198b-03f3-
00151725d513/VM_Name2/VM_Name2-000001-delta.vmdk" : open successful (23) size =
83886080, hd = 0. Type 8
DISKLIB-LINK : Attach: the capacity of each link is different (163840 != 1163840).
DISKLIB-CHAIN : "/vmfs/volumes/4b72a71c-0047198b-03f3-
00151725d513/VM_Name2/VM_Name2-000001.vmdk" : failed to open (The parent virtual
disk has been modified since the child was created).
DISKLIB-VMFS : "/VM_Name2-000002-delta.vmdk" : closed.
DISKLIB-VMFS : "/vmfs/volumes/4b72a71c-0047198b-03f3-
00151725d513/VM_Name2/VM_Name2-000001-delta.vmdk" : closed.
DISKLIB-LIB : Failed to open 'VM_Name2-000002.vmdk' with flags 0x17 (The parent
virtual disk has been modified since the child was created).
Failed to open 'VM_Name2-000002.vmdk' : The parent virtual disk has been modified
since the child was created (18).
```

Explanation

DISKLIB-LINK : Attach: the capacity of each link is different (163840 != 1163840).	The RW numbers do not match
DISKLIB-CHAIN : "/vmfs/volumes/4b72a71c-0047198b-03f3-00151725d513/VM_Name2/VM_Name2-000001.vmdk"	Between this file and its child

How can this happen? Because someone has expanded the size of the vDisk of the VM when it had snapshots, which is unsupported.

A virtual machine cannot boot after extending a base virtual disk that is part of a snapshot hierarchy (<http://kb.vmware.com/kb/1646892>).

2.4.3.3. Finding missing vmdk files with vmkfstools

No issue

```
# vmkfstools -q test_2-000002.vmdk
test_2-000002.vmdk is not an rdm
```

One vmdk descriptor in the chain has been deleted

```
# vmkfstools -q test_2-000002.vmdk
Failed to open 'test_2-000002.vmdk' : The parent of this virtual disk could not be
opened (23).
```

With verbose output

```
# vmkfstools -q test_2-000002.vmdk -v 1
DISKLIB-VMFS : "/test_2-000002-delta.vmdk" : open successful (23) size = 73400320,
hd = 0. Type 8
DISKLIB-LINK : "/vmfs/volumes/4b72a71c-0047198b-03f3-00151725d513/test2/test_2-
000001.vmdk" : failed to open (The system cannot find the file specified).
DISKLIB-CHAIN : "test_2-000002.vmdk": Failed to open parent "/vmfs/volumes/4b72a71c-
```

```
0047198b-03f3-00151725d513/test2/test_2-000001.vmdk": The system cannot find the
file specified.
DISKLIB-CHAIN : "/vmfs/volumes/4b72a71c-0047198b-03f3-00151725d513/test2/test_2-
000001.vmdk" : failed to open (The parent of this virtual disk could not be opened).
DISKLIB-VMFS : "./test_2-000002-delta.vmdk" : closed.
DISKLIB-LIB : Failed to open 'test_2-000002.vmdk' with flags 0x17 (The parent of
this virtual disk could not be opened).
Failed to open 'test_2-000002.vmdk' : The parent of this virtual disk could not be
opened (23).
```

Explanation

DISKLIB-LINK : "/vmfs/volumes/4b72a71c-0047198b-03f3-00151725d513/test2/test_2-000001.vmdk" : failed to open (The system cannot find the file specified).	This is the missing file
-----------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------

Now, a delta vmdk has been deleted

```
# vmkfstools -q test_2-000002.vmdk -v 1
DISKLIB-VMFS : "./test_2-000002-delta.vmdk" : open successful (23) size = 73400320,
hd = 0. Type 8
DISKLIB-VMFS : "/vmfs/volumes/4b72a71c-0047198b-03f3-00151725d513/test2/test_2-
000001-delta.vmdk" : failed to open (25): Backing file doesn't exist. Type 8
DISKLIB-DSCPTR: Failed to open extents for descriptor file in normal mode
DISKLIB-LINK : "/vmfs/volumes/4b72a71c-0047198b-03f3-00151725d513/test2/test_2-
000001.vmdk" : failed to open (The system cannot find the file specified).
DISKLIB-CHAIN : "test_2-000002.vmdk": Failed to open parent "/vmfs/volumes/4b72a71c-
0047198b-03f3-00151725d513/test2/test_2-000001.vmdk": The system cannot find the
file specified.
DISKLIB-CHAIN : "/vmfs/volumes/4b72a71c-0047198b-03f3-00151725d513/test2/test_2-
000001.vmdk" : failed to open (The parent of this virtual disk could not be opened).
DISKLIB-VMFS : "./test_2-000002-delta.vmdk" : closed.
DISKLIB-LIB : Failed to open 'test_2-000002.vmdk' with flags 0x17 (The parent of
this virtual disk could not be opened).
Failed to open 'test_2-000002.vmdk' : The parent of this virtual disk could not be
opened (23).
```

Explanation

DISKLIB-VMFS : "/vmfs/volumes/4b72a71c-0047198b-03f3-00151725d513/test2/test_2-000001-delta.vmdk" : failed to open (25): Backing file doesn't exist. Type 8	This is the missing file
-------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------

2.4.4. Hostd behaving abnormally

Sometimes what you are trying to do is failing just because the *hostd* process is in an 'unstable/hung' state. You may need to restart/kill it before you can continue troubleshooting the snapshots problem.

Restarting the Management agents on an ESX or ESXi Server (<http://kb.vmware.com/kb/1003490>)

If you see that in that host you don't make any progress, you can VMotion (hot or cold migration) the VM to another host and try from there.

2.4.5. Check KB

If you get/see strange error messages on the service console or on the logs, it is worth taking a look to the [KB](#).

2.5. Other options

2.5.1. Clone the VM

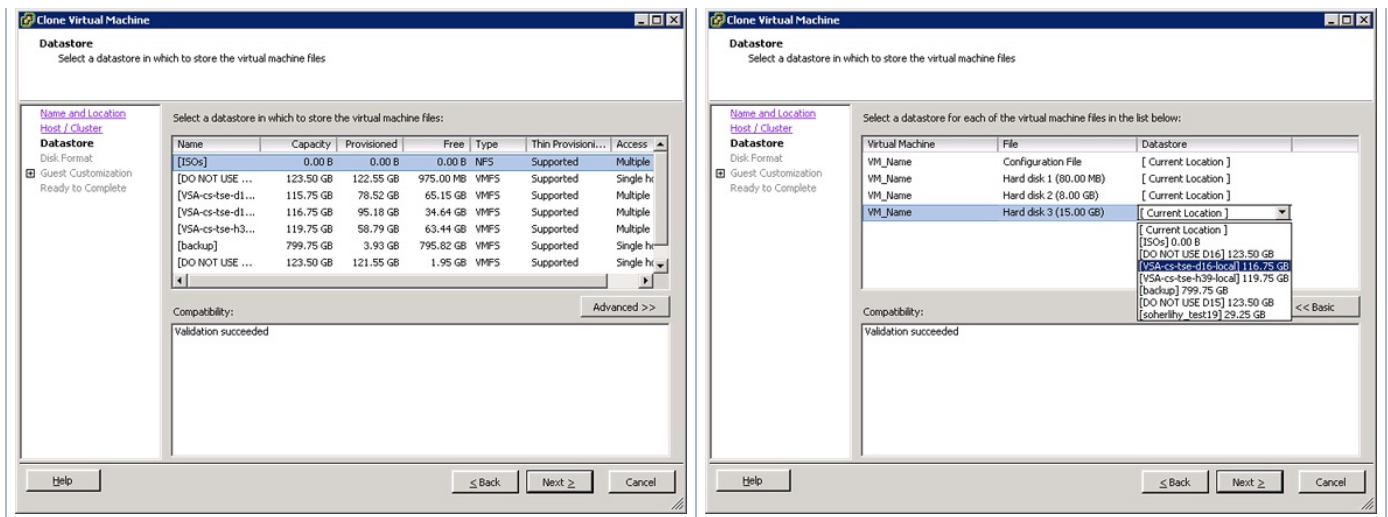
If you clone the VM, the clone will have the same information as the source VM but it will have no snapshots, i.e. in the cloning process the snapshots will be committed into the cloned virtual disks.

This option is handy when the VM has several snapshots and not very big Base Disks and there is free space somewhere else to put the clone there.

Note: You need to connect the VIClient to the Virtual Center to be able to clone a VM.

If the VM is running while you clone it, an additional snapshot will be created.

Notice that you don't necessarily need the total space of the VM available in a single Datastore. If on the third step of the cloning wizard you click on "Advanced" you will be able to select where you want to put every piece of the VM. Additionally the cloning process may complete quicker if you separate the disks as the I/O will go to different Datastores in parallel



2.5.2. Clone virtual disks

As well as cloning a VM, cloning a disk with snapshots will commit the snapshots into the destination virtual disk. Just be careful to point at the last snapshot on the chain you want to commit.

Note: If you want to clone the whole chain, the VM must be powered off. You cannot clone a disk that is being used.

The command to clone a virtual disk, with or without snapshots is:

```
# vmkfstools -i Seed.vmdk /Path/to/destination/Clone_Name.vmdk
```

Notice that you must always point at the .vmdk descriptor file.

- Let's see some examples:

Imagine the following chain of snapshots:

```
VM_Name-000004.vmdk -> VM_Name-000005.vmdk -> VM_Name-000003.vmdk -> VM_Name-000001.vmdk -> VM_Name.vmdk
```

If you run:

```
# vmkfstools -i VM_Name-000004.vmdk VM_Name.Clone.all.vmdk
```

The result will be:

```
VM_Name-000004.vmdk +
VM_Name-000005.vmdk +
VM_Name-000003.vmdk +
VM_Name-000001.vmdk +
VM_Name.vmdk +
-----
VM_Name.Clone.all.vmdk
```

And if you do:

```
# vmkfstools -i VM_Name-000003.vmdk VM_Name.Clone.2.vmdk
```

The result will be:

```
VM_Name-000003.vmdk +
VM_Name-000001.vmdk +
VM_Name.vmdk +
-----
VM_Name.Clone.2.vmdk
```

Warning: Try to create the clones with very descriptive names. It is a pain to take a case where someone created some clones before and discover that they all have the same name. You don't know what contains what.

Warning: Try to avoid using spaces or strange symbols on the names. Just use 0..9a..Z_-

2.5.3. Commit one virtual disk at a time

If you don't have space to commit all the disks of the VM but you can afford to commit one chain at a time then do the following (example):

1. Note the disks attached to the VM and save this information somewhere as it will be used later.

```
# sgrep VMName.vmx
scsi0:0.present = "true"
scsi0:0.fileName = "VMName-000006.vmdk"
scsi0:1.present = "true"
scsi0:1.fileName = "VMName_1-000006.vmdk"
scsi0:2.present = "true"
scsi0:2.fileName = "VMName_2-000006.vmdk"
scsi0:3.present = "true"
scsi0:3.fileName = "VMName_3-000006.vmdk"
scsi1:0.present = "false"
scsi1:0.fileName = "/vmfs/volumes/4688d0e7-7b5c822c-61d7-00145e808070/VMName/VMName_4.vmdk"
```

2. Edit the .vmx and set all of the `true` disks but one to `false` and verify.

```
# sgrep VMName.vmx
scsi0:0.present = "true"
scsi0:0.fileName = "VMName-000006.vmdk"
scsi0:1.present = "false"
scsi0:1.fileName = "VMName_1-000006.vmdk"
scsi0:2.present = "false"
scsi0:2.fileName = "VMName_2-000006.vmdk"
scsi0:3.present = "false"
scsi0:3.fileName = "VMName_3-000006.vmdk"
scsil:0.present = "false"
scsil:0.fileName = "/vmfs/volumes/4688d0e7-7b5c822c-61d7-00145e808070/VMName/VMName_4.vmdk"
```

3. Follow the procedure of [committing snapshots when there are no snapshot entries in the snapshot manager](#) to commit that chain.
4. Check

```
# sgrep VMName.vmx
scsi0:0.present = "true"
scsi0:0.fileName = "VMName.vmdk"
scsi0:1.present = "false"
scsi0:1.fileName = "VMName_1-000006.vmdk"
scsi0:2.present = "false"
scsi0:2.fileName = "VMName_2-000006.vmdk"
scsi0:3.present = "false"
scsi0:3.fileName = "VMName_3-000006.vmdk"
scsil:0.present = "false"
scsil:0.fileName = "/vmfs/volumes/4688d0e7-7b5c822c-61d7-00145e808070/VMName/VMName_4.vmdk"
```

5. Edit again the .vmx setting all but the next disk to `false`.

```
# sgrep VMName.vmx
scsi0:0.present = "false"
scsi0:0.fileName = "VMName.vmdk"
scsi0:1.present = "true"
scsi0:1.fileName = "VMName_1-000007.vmdk"
scsi0:2.present = "false"
scsi0:2.fileName = "VMName_2-000006.vmdk"
scsi0:3.present = "false"
scsi0:3.fileName = "VMName_3-000006.vmdk"
scsil:0.present = "false"
scsil:0.fileName = "/vmfs/volumes/4688d0e7-7b5c822c-61d7-00145e808070/VMName/VMName_4.vmdk"
```

6. Commit that chain and verify

```
# sgrep VMName.vmx
scsi0:0.present = "false"
scsi0:0.fileName = "VMName.vmdk"
scsi0:1.present = "true"
scsi0:1.fileName = "/vmfs/volumes/482c6a32-da3cdd8a-646a-001a4baf5986/VMName/VMName_1.vmdk"
scsi0:2.present = "false"
scsi0:2.fileName = "VMName_2-000006.vmdk"
scsi0:3.present = "false"
scsi0:3.fileName = "VMName_3-000006.vmdk"
scsil:0.present = "false"
scsil:0.fileName = "/vmfs/volumes/4688d0e7-7b5c822c-61d7-00145e808070/VMName/VMName_4.vmdk"
```

7. Repeat the process with all the disks that were `true` on the step number 1 until all their chains are committed.

```
# sgrep VMName.vmx
scsi0:0.present = "false"
scsi0:0.fileName = "VMName.vmdk"
scsi0:1.present = "false"
scsi0:1.fileName = "/vmfs/volumes/482c6a32-da3cdd8a-646a-001a4baf5986/VMName/VMName_1.vmdk"
scsi0:2.present = "false"
scsi0:2.fileName = "/vmfs/volumes/4688d0e7-7b5c822c-61d7-00145e808070/VMName/VMName_2.vmdk"
scsi0:3.present = "true"
scsi0:3.fileName = "/vmfs/volumes/47b0960d-268c7dd0-e4d5-001a4baf5986/VMName/VMName_3.vmdk"
scsil:0.present = "false"
scsil:0.fileName = "/vmfs/volumes/4688d0e7-7b5c822c-61d7-00145e808070/VMName/VMName_4.vmdk"
```

8. Now set the status of the disks back to `true` according with the mapping of step 1.

```
# sgrep VMName.vmx
scsi0:0.present = "true"
scsi0:0.fileName = "VMName.vmdk"
scsi0:1.present = "true"
scsi0:1.fileName = "/vmfs/volumes/482c6a32-da3cdd8a-646a-001a4baf5986/VMName/VMName_1.vmdk"
scsi0:2.present = "true"
scsi0:2.fileName = "/vmfs/volumes/4688d0e7-7b5c822c-61d7-00145e808070/VMName/VMName_2.vmdk"
scsi0:3.present = "true"
scsi0:3.fileName = "/vmfs/volumes/47b0960d-268c7dd0-e4d5-001a4baf5986/VMName/VMName_3.vmdk"
scsil:0.present = "false"
```

```
scsi1:0.fileName = "/vmfs/volumes/4688d0e7-7b5c822c-61d7-00145e808070/VMName/VMName_4.vmdk"
```

2.5.4. Commit manually to the base disk

As you have seen [before](#), committing a snapshot directly to the Base Disk requires no additional space. This may be the only solution when you have no space to commit the chain of a single Base Disk and there are no other datastores usable for cloning.

So if you have something like this:

Snap2 -> Snap1 -> BaseDisk

```
# sgrep VM_Name.vmx
scsi0:0.present = "TRUE"
scsi0:0.fileName = "VM_Name-000002.vmdk"

# sgrep VM_Name-000002.vmdk
CID=abd2fcd6
parentCID=abd2fcd6
parentFileNameHint="VM_Name-000001.vmdk"
RW 163840 VMFSSPARSE "VM_Name-000002-delta.vmdk"

# sgrep VM_Name-000001.vmdk
CID=abd2fcd6
parentCID=abd2fcd6
parentFileNameHint="VM_Name.vmdk"
RW 163840 VMFSSPARSE "VM_Name-000001-delta.vmdk"
```

You can commit the chain doing this very manual process:

1. Point the VM disk to Snap1

```
### BEFORE
# sgrep VM_Name.vmx
scsi0:0.present = "TRUE"
scsi0:0.fileName = "VM_Name-000002.vmdk"

### AFTER
# sgrep VM_Name.vmx
scsi0:0.present = "TRUE"
scsi0:0.fileName = "VM_Name-000001.vmdk"
```

2. Follow the [standard procedure](#) to commit the chain *Snap1 -> BaseDisk*.

```
# vmware-cmd VM_Name.vmx createsnapshot ToRemove
createsnapshot(ToRemove) = 1

# sgrep VM_Name.vmx
scsi0:0.present = "TRUE"
scsi0:0.fileName = "VM_Name-000003.vmdk"

# vmware-cmd VM_Name.vmx removesnapshots
removesnapshots() = 1

# sgrep VM_Name.vmx
scsi0:0.present = "TRUE"
scsi0:0.fileName = "VM_Name.vmdk"
```

3. Modify *parentCID* and *parentFileNameHint* on Snap2 to make it point at *BaseDisk*'.

```
### BEFORE
# sgrep VM_Name-000002.vmdk
CID=abd2fcd6
parentCID=abd2fcd6
parentFileNameHint="VM_Name-000001.vmdk"
RW 163840 VMFSSPARSE "VM_Name-000002-delta.vmdk"

### AFTER
# sgrep VM_Name-000002.vmdk
CID=abd2fcd6
parentCID=abd2fcd6
parentFileNameHint="VM_Name.vmdk"
RW 163840 VMFSSPARSE "VM_Name-000002-delta.vmdk"
```

4. Point the VM disk at Snap2

```
### BEFORE
# sgrep VM_Name.vmx
scsi0:0.present = "TRUE"
scsi0:0.fileName = "VM_Name.vmdk"

### AFTER
# sgrep VM_Name.vmx
scsi0:0.present = "TRUE"
scsi0:0.fileName = "VM_Name-000002.vmdk"
```

5. Commit the chain *Snap2 -> BaseDisk*'

```
# vmware-cmd VM_Name.vmx createsnapshot ToRemove
createsnapshot(ToRemove) = 1
# vmware-cmd VM_Name.vmx removesnapshots
removesnapshots() = 1
```

With this procedure you can remove any number of snapshots. However it has one requirement, you need at least 16MB of

space on the datastore in order to create the snapshot that you will use to remove the snapshot above the BaseDisk.

2.5.5. Other options

In some cases, using VMware Converter is a very good solution. You can 'convert' the VM while it is running in a brand new VM without snapshots. However it has some requirements:

- You need to install VMware Converter on the VM.
- The VM has to be running during the conversion.
- You need free space somewhere (local or remote Datastore) to copy a clone of the VM.

2.6. How can we free up space on the Datastore?

There are several ways, some are easier and faster than others, so you will have to evaluate your options. Here are some:

- Find other VMs with snapshots on the same datastore that can be deleted. See [useful commands](#) to know how.
- Relocate the disk(s) of other VMs with *cold migration* to another Datastore. **Warning!** Do not migrate disks with snapshots.
- Relocate templates.
- Delete something unnecessary. Browse the Datastore to know what is inside.

2.7. General important considerations

- If the Datastore is full and there is at least one VM running on snapshots in it, the VM will crash soon, shut it down before it happens!
- If you or the customer is unsure about the content of the snapshots or the consistency of the information, it is wise to create a clone disk and work on it so the original files stay untouched. This will allow you to come back to them later if needed.
- If you delete a snapshot from VC and the operation takes more than 15 minutes, it will appear as 'timed out' even if it is still working on the background. You will not experience timeout if you delete the snapshot from the VI Client connected directly to the ESX. Also, if during the commit operation you open a VI Client into the ESX you will see there the task running.
- If the customer says that he clicked on 'Delete all' and he still sees snapshots on the snapshot manager, check if the task is [still in progress](#). The VM will consider that it has snapshots until the task is completed or something goes wrong.

2.7.1. Extra considerations

As mentioned previously this guide was designed for ESX 3.5. For other versions some things may be slightly different. The details below are not the complete list.

2.7.1.1. ESXi 3.5 and 4.0

There is no /var/log/vmkernel Instead you have /var/log/messages which contains entries for vmkernel and hostd together.

```
Mar  2 18:00:03 Hostd: .....
Mar  2 18:00:10 vmkernel: .....
```

In this version using the Service Console is not supported, be aware. The "correct" way to send commands to an ESXi is using RCLI (Remote CLI) [\[6\]](#)

If you want to go for the unsupported way, here are the minimum commands you will need.

1. Find the VM id (vmid)

```
## Get the list of virtual machines on the host.
## Usage: getallvms

# vim-cmd vmshvc/getallvms
Vmid  Name      File                                     Guest OS
Version  Annotation
1184   VM_Name2    [VSA-cs-tse-d15-local] VM_Name2/VM_Name2.vmx  rhel5Guest vmx-07
```

2. Check if it says it has snapshots

```
## Gets the snapshot info for the vm.
## Usage: snapshot.get vmid

# vim-cmd vmshvc/snapshot.get 1184
Get Snapshot:
```

3. If not, create one

```
## Creates a snapshot for the vm.
## Usage: snapshot.create vmid [snapshotName] [snapshotDescription]
[includeMemory] [quiesced]

# vim-cmd vmshvc/snapshot.create 1184 SnapName SnapDesc 0 0
Create Snapshot:
```

4. Verify it has been successfully created

```
# vim-cmd vmshvc/snapshot.get 1184
Get Snapshot:
|-ROOT
--Snapshot Name      : SnapName
--Snapshot Description : SnapDesc
--Snapshot Created On  : 3/2/2010 19:53:29
--Snapshot State     : powered off
```

5. And now removeall

```
## Removes all the snapshots on the vm.
```

```
## Usage: snapshot.removeall vmid  
  
# vim-cmd vmsvc/snapshot.removeall 1184  
Remove All Snapshots:
```

2.7.1.2. ESX 4.0

In ESX4 this command has the same format, but now all the arguments are expected.

```
## FORMAT  
/usr/bin/vmware-cmd <cfg> createsnapshot <name> <description> <quiesce> <memory>  
  
## WRONG  
# vmware-cmd VM_Name.vmx createsnapshot Name Desc  
TypeError: DoIt() takes exactly 7 arguments (5 given)  
  
## CORRECT (no quiesce, no memory)  
# vmware-cmd VM_Name.vmx createsnapshot Name Desc 0 0  
createsnapshot(Name, Desc, 0, 0) = 1
```

2.7.1.3. ESXi 4.0 and ESX 4.0

There is a very small detail that could invalidate some of the premises of this guide and therefore make the rest of the content not applicable: In ESX4 you can use thin vDisks. Their size is not preallocated, which means that they grow with the content they store. When a snapshot commits its content into a thin vDisk the thin vDisk will grow if the snapshot contains information located in sectors that haven't been used already by the Base Disk.

3. Related KBs

3.1. Public

Virtual Machine Cannot Boot After Extending a Base Virtual Disk That Is Part of Snapshot Hierarchy
(<http://kb.vmware.com/kb/1646892>)

Delete All Snapshot Operation Results in Consolidate Helper Snapshot When Datastore Has Insufficient Disk Space
(<http://kb.vmware.com/kb/1003302>)

Committing snapshots from within the Service Console (<http://kb.vmware.com/kb/1006847>)

Committing snapshots when there are no snapshot entries in the snapshot manager (<http://kb.vmware.com/kb/1002310>)

Why snapshot removal can stop a virtual machine for long time (<http://kb.vmware.com/kb/1002836>)

Microsoft Exchange Server on a Virtual Machine Can Freeze Under Load When You Take Quiesced Snapshots or Use Custom Quiescing Scripts (<http://kb.vmware.com/kb/5962168>)

Consolidating snapshots (<http://kb.vmware.com/kb/1007849>)

If a Virtual Machine Has Two Disks with Identical File Names, Stored in Different Locations, You Cannot Delete a Snapshot of This Virtual Machine (<http://kb.vmware.com/kb/5096672>)

Virtual Disk Remains on a Datastore After Snapshots Referencing the Disk Are Deleted (<http://kb.vmware.com/kb/1003002>)

Increasing the Size of the Hard Disk Fails on a Virtual Machine That Has Snapshots (<http://kb.vmware.com/kb/1003099>)

Snapshot Operations Submitted Directly to an ESX Server Host During Storage VMotion Corrupt Virtual Machine Data
(<http://kb.vmware.com/kb/1003114>)

Deleting Snapshots of Virtual Machines With Heavy Disk I/O Might Cause Host to Be Disconnected from VirtualCenter
(<http://kb.vmware.com/kb/1003024>)

After cold migration on ESX Server, virtual disk with snapshot has the wrong CID (<http://kb.vmware.com/kb/1005228>)

Committing snapshots generates a content ID mismatch error (<http://kb.vmware.com/kb/1007969>)

Consolidation of Large or Deeply Nested Snapshots Using VirtualCenter, SDK, or VCB Might Take Longer on ESX Server 3.5 than on ESX Server 3.0.x (<http://kb.vmware.com/kb/1003308>)

Troubleshooting snapshot commit issue on a virtual machine (<http://kb.vmware.com/kb/1004538>)

Taking More Than One Snapshot of a Suspended Virtual Machine Produces an Error (<http://kb.vmware.com/kb/1424776>)

Virtual Machine Changes to Snapshot-Consolidate Mode During Delete-All-Snapshot Operation
(<http://kb.vmware.com/kb/2222>)

Cannot Undo Revert Snapshot (<http://kb.vmware.com/kb/1000176>)

Taking a Snapshot of a Virtual Machine with Independent, Non-Persistent Disks Fails (<http://kb.vmware.com/kb/1001556>)

Reverting to Snapshots of Virtual Machines with Hot-Added RDM Virtual Disks in Physical or Virtual Mode Causes Virtual Machines to Hang (<http://kb.vmware.com/kb/9729125>)

"Unable to save snapshot file" Error Occurs When Taking a Snapshot (<http://kb.vmware.com/kb/1000310>)

Unable to Delete Snapshots After Restore From Backup (<http://kb.vmware.com/kb/1002726>)

Verifying ESX virtual machine file integrity (<http://kb.vmware.com/kb/1003743>)

Verifying sufficient free disk space for an ESX virtual machine (<http://kb.vmware.com/kb/1003755>)

Adding Space to an ESX Server Virtual Disk (<http://kb.vmware.com/kb/994>)

Increasing the size of a virtual disk (<http://kb.vmware.com/kb/1004047>)

Cannot power on a virtual machine because the virtual disk cannot be opened (<http://kb.vmware.com/kb/1004232>)

Adding and committing redo logs (<http://kb.vmware.com/kb/1004458>)

Verifying the integrity of the parent disks for an ESX virtual machine (<http://kb.vmware.com/kb/1003759>)

3.2. Patches

VMware ESX 3.5, Patch ESX350-200912401-BG: Updates VMkernel, Tools, hostd (<http://kb.vmware.com/kb/1016657>)

VMware ESX 3.5, Patch ESX350-200901401-SG: Updates VMkernel, VMX, and hostd (<http://kb.vmware.com/kb/1006651>)

VMware ESX 3.5, Patch ESX350-200804402-BG: Update to VMware-esx-vmx (<http://kb.vmware.com/kb/1004162>)

VMware ESX 3.5, Patch ESX350-200901401-SG: Updates VMkernel, VMX, and hostd (<http://kb.vmware.com/kb/1006651>)

VMware ESXi, Patch ESXi350-200901401-ISG: Firmware Update (<http://kb.vmware.com/kb/1006661>)

VMware ESX Server 3.5, Patch ESXi350-200804401-O-BG: Firmware Update (<http://kb.vmware.com/kb/1004169>)

VMware ESX Server 3.5, Patch ESX350-200804402-BG: Update to VMware-esx-vmx (<http://kb.vmware.com/kb/1004162>)

ESX Server 3.0.2, Patch ESX-1004216: VMkernel LVM driver Might Stop Responding; VMware VIX API Memory Overflow Vulnerabilities; Snapshot Operations Might Fail Under High I/O Stress (<http://kb.vmware.com/kb/1004216>)

ESX Server 3.0.2, Patch ESX-1004210: Virtual Machine Migration; Thumbnail Files Accumulate; SMTP Data Packets Get Dropped; Virtual Machine Does Not Synchronize; ESX Server Host Might ASSERT or Not Respond During Snapshot Operations (<http://kb.vmware.com/kb/1004210>)

ESX Server 3.0.2, Patch ESX-1002088: vpxa Upgrade Failure; Internationalized ESX Server 3.x Product Home Page; Update to Allow Snapshot with RDM Disks Attached; Other VMware-hostd-esx.rpm Updates (<http://kb.vmware.com/kb/1002088>)

ESX Server 3.0.1, Patch ESX-8258730: Security Updates, Updates to Monitor and LSI Logic SCSI Emulator Driver (<http://kb.vmware.com/kb/8258730>)

ESX Server 3.0.1, Patch ESX-1004822: Snapshot Operations Might Fail Under High I/O Stress (<http://kb.vmware.com/kb/1004822>)

ESX Server 3.0.1, Patch ESX-1004185: Thumbnail Files Accumulate in ESX Server Host; Virtual Machine Does Not Synchronize With ESX Server Host Time; ESX Server Host Might ASSERT or Not Respond During Snapshot Operations (<http://kb.vmware.com/kb/1004185>)

ESX Server 3.0.1, Patch ESX-1002083: vpxa Upgrade Failure; Update to Allow Snapshot with RDM Disks Attached; Other VMware-hostd-esx.rpm Updates (<http://kb.vmware.com/kb/1002083>)

ESX Server 3.0.1, Patch ESX-1000077: Update to Host Agent (<http://kb.vmware.com/kb/1000077>)

ESX Server 3.0.0, Patch ESX-4809553: Security Updates; Updates to Monitor (<http://kb.vmware.com/kb/4809553>)

ESX Server 3.0.0, Patch ESX-1000081: Update to Host Agent (<http://kb.vmware.com/kb/1000081>)

4. Preventing issues with snapshots

There are several proactive measures you can take to avoid having problems with snapshots.

- Install the patches [\[4\]](#) related with snapshots.
- Do not let the Snapshots grow too big [\[5\]](#).
- **Do not allow the Service Console to swap.** This is the root of many issues.

Increasing the amount of RAM assigned to the ESX Server service console (<http://kb.vmware.com/kb/1003501>)

You may wonder, How do I know if the Service Console is swapping?

Run the command "free -m"

#	free -m						
		total	used	free	shared	buffers	cached
Mem:		291	229	62	0	36	136
-/+ buffers/cache:			56	235			
Swap:		596	XX	535			

The number on XX is the amount of Megabytes that the Service Console is using or has used from the swap space. It should be zero.

5. Knowledge gained from experience

- A VM with many large Base Disks (hundreds of Gigabytes in size) and several large snapshots (tens or hundreds of GBs in size) is like a transatlantic ship. Once its moving it can go fast, but taking it out of port take ages. A VM with these characteristics may even take up to 30 minutes to power on.
- If you have to commit very large snapshots (tens or hundreds of GBs in size), it is better to schedule some downtime and do it with the VM down and even better, out of business hours, as doing it with the VM running may reduce its performance quite a lot or even crash it.
- If you have weird problems removing the snapshots created by a 3rd party Backup Software, turn the Backup server off if you are not inside the Backup window. It may have some processes opened against the VM that will make you unable to create/remove snapshots.
- Do not panic if you see a snapshot file that is slightly bigger than the Base Disk. That extra space is used by the data structures that the snapshot need to arrange the data it contains.
- If once you have started committing large snapshots you loose VIClient access to the host (SSH stays alive) do not worry. When the operation is completed you will get access back. If you don't like this, remember what I said about [do not allow the Service Console to swap](#) and [install the patches](#).

6. Frequently asked questions

6.1. How long is it going to take to delete all the snapshots?

It depends on factors such as:

- Storage speed
- How busy the ESX/Storage is
- How big the snapshots are and how many of them there are
- If the VM is running/off

In any case, there is no answer to this question. You can only [watch](#) the evolution of the committing operation and from there you may be able to make an estimation.

6.2. How do I know when the 'removesnapshots' operation is completed?

You should see something like this on the Service Console:

```
# vmware-cmd VM_Name.vmx removesnapshots
removesnapshots() = 1
```

In the VI Client connected directly to the ESX you should see when the task is completed.

Recent Tasks						
Name	Target	Status	Initiated by	Time	Start Time	Complete Time
Logout		Completed	root	21/01/2009 16:41:11	21/01/2009 16:41:11	21/01/2009 16:41:11
Remove All Snapshots	VM_Name	Completed	root	21/01/2009 16:41:09	21/01/2009 16:41:09	21/01/2009 16:41:11
Logout		Completed	root	21/01/2009 16:40:41	21/01/2009 16:40:41	21/01/2009 16:40:41

6.3. How do I know that the 'removesnapshots' operation is really working?

Using [watch](#) on the VM folder. Alternatively you can use `vimsh` (see [\[2\]](#) for more info) to check if the task is completed.

See here an example where we query about the tasks for a specific VM. One task appears listed, however it has been completed already.

```
# vmware-vim-cmd vmsvc/getallvms
Vmid      Name      File
Guest OS
288      bf_rhel5      [bf_Ubuntu_nfs-1] bf_rhel5/bf_rhel5.vmx
rhel5Guest      vmx-04
304      VM_Name      [ds-vol2] VM_Name/VM_Name.vmx
winNetStandar st      vmx-04
320      AR-RHEL5-01 [hl6-local1] AR-RHEL5-01/AR-RHEL5-01.vmx
rhel5Guest      vmx-04

# vmware-vim-cmd vmsvc/get.tasklist 304
(ManagedObjectReference) [
  'vim.Task:haTask-304-vim.VirtualMachine.removeAllSnapshots-39439'
]

# vmware-vim-cmd vmsvc/task_info 'vim.Task:haTask-304-
vim.VirtualMachine.removeAllSnapshots-39439'
(vmodl.fault.ManagedObjectNotFound) {
  dynamicType = <unset>,
  obj = 'vim.Task:vim.Task:haTask-304-vim.VirtualMachine.removeAllSnapshots-39439',
  msg = "The request refers to an object that no longer exists or has never
existed."
}
```

Also, in the VI Client connected directly to the ESX you should see the task in progress (probably stuck at 95%, but that is normal).

7. Useful commands

- You can find all the snapshots on all the datastores visible from one host running this on the service console:

```
find /vmfs/volumes/ -name "*delta*" -type f -print0 | xargs -0 du --human-
readable --total
```

- Useful to see only the lines that you need from a .vmx or .vmdk file.

```
alias sgrep='egrep -i --color "scsi[0-9]+:[0-
9]+.present|scsi.*filename|vmdk|parent|CID"'
```

- To see the space available on all the Datastores

```
vdf -h
```

- To see the space available in the Datastore we are located

```
vdf -h .
```

- Is there a missing file? Look for it in all the Datastores.

```
find /vmfs/volumes/ -iname "*FILENAME*"
```

- To see the evolution of the modifications on the files of the VM

```
watch -d "ls -lt *.vmdk ;date"
```

- To have a nice colorful prompt

```
PS1='\[\e[0;31m\]\u@\h \W]$ \[\e[m\]\[\e[0;32m\]' ; export PS1
```

8. Acknowledgement

The author would like to thank Darren Burnett and Antonio Allegue Leira for their revisions of this document.

The author would also like to thank Edina Varallyai for her support while modelling mathematically the snapshot consolidation process.

9. References

- [1] ESX 3.5 Basic System Administration
http://www.vmware.com/pdf/vi3_35/esx_3/r35u2/vi3_35_25_u2_admin_guide.pdf
 - [2] What Files Make Up a Virtual Machine? http://www.vmware.com/support/ws5/doc/ws_learning_files_in_a_vm.html
 - [3] VIMSH for ESX 3.5 <http://knowledge.xtravirt.com/white-papers> , vimsh
http://www.engr.ucsb.edu/~duonglt/vmware/#vmware_vimsh
 - [4] Patches for VI3 (<https://www.vmware.com/mysupport/download/allpatches.html>) and vSphere
(<http://tinyurl.com/ycakr8z>)
 - [5] For vCenter Server only "Configuring VMware vCenter Server to send alarms when virtual machines are running from snapshots (<http://kb.vmware.com/kb/1018029>)"
 - [6] vSphere Command-Line Interface <http://www.vmware.com/support/developer/vcli/>
-