

## Initializing Windows Disks with DiskPart

This oft-forgotten tool is indispensable when you're building systems



Mark Minasi

One of Windows' essential command-line utilities is Diskpart, the successor to Fdisk that appeared first in Windows XP. The tool lets you partition, format, initialize, and resize drives, set up RAID, and more—and in Windows 7 and Windows Server 2008 R2, it lets you work with virtual disks. I find Diskpart indispensable when building systems, so I'm often surprised to hear that many IT pros don't know much about it. So, let's dig in and see how to use Diskpart to wipe a hard disk clean, partition the disk, and format it.

At the command line, type `diskpart` and press Enter. Once you're in the tool, you'll Diskpart has its own command prompt. What you're seeing is that Diskpart is essentially its own command environment—sort of a command line interface (CLI) inside a CLI. Typing `help` (or any invalid command) and pressing Enter causes Diskpart to display about three dozen commands, but don't let the prospect of learning 37 commands scare you away from Diskpart because you'll actually perform 99 percent of your Diskpart work with about eight commands.

As you know, if you've ever initialized a disk from the Logical Disk Manager GUI, you first click on the physical disk drive, then create a partition (or partitions) on it, then format those partitions and typically give them letters. Diskpart follows the same pattern, requiring you to select a disk before you create partitions on it, then requiring you to select a partition before formatting it, and so on. To select a particular disk, you type a command that looks like

```
select disk <disknumber>
```

but which number is the disk you want to work on? Find out by typing `list disk`, which gives you output that looks like Figure 1. Diskpart numbers disks starting from zero rather than one, and you can see that both drives store their partition information on the master boot record (MBR) rather than a GUID partition table (GPT), and both disks are basic rather than dynamic (no values in Dyn and Gpt columns).

Disk ###	Status	Size	Free	Dyn	Gpt
Disk 0	Online	24 GB	1024 KB		
Disk 1	Online	24 GB	0 B		

Figure 1: List Disk Output

Now, I'll complicate the problem by saying that this system has two 24GB drives—one holds the OS and applications, and the other is just an extra drive for holding data. Clearly, you don't want to accidentally wipe the OS's drive, so how do you know which is which? You use `select disk`, along with `detail disk`. Let's start with disk 0 and see what's on it:

```
select disk 0  
  
detail disk
```

```

VMware Virtual IDE Hard Drive ATA Device
Disk ID           : F1B17FB3
Type              : ATA
Status           : Online
Path             : \
Target           : \
LUN ID           : 0
Location Path    : PCIROOT(0)#PCI(0701)#ATA(C00T00L00)
Current Read-only State : No
Read-only        : No
Boot Disk        : No
Pagefile Disk    : No
Hibernation File Disk : No
Crashdump Disk   : No
Clustered Disk   : No

Volume ### Ltr Label      Fs      Type          Size  Status      Info
-----
Volume 1   E   New Volume NTFS     Partition 23 GB Healthy

```

Figure 2: Detail Disk Output

That provides output that looks like Figure 2. Disk 0, then, is—surprisingly—the data disk, and therefore disk 1 must be the OS disk. So, you've got your sights on the correct disk (disk 0). Next, let's wipe the disk clean by typing *clean*. Doing so doesn't really wipe a disk clean by overwriting all of its data; rather, *clean* erases the MBR, which is essentially the disk's "table of contents." The old data is still on the disk, but the OS doesn't know how to get to it anymore, so that data will end up being slowly overwritten as we build a new disk structure atop the old one and start putting files on that new disk structure.

Next, the newly cleaned disk needs at least one new disk partition. The simplest method is to make the drive into one big partition with command *create partition primary*. If, however, you want to chop the drive into multiple partitions, you'd add the *size=* parameter, followed by the desired size in megabytes. If I wanted to create two partitions—one 10GB and another 14GB—I'd create the first one by typing

```
create partition primary size=10240
```

(Remember, there are 1024MB in a gigabyte, not 1000.) Then, I could create the second partition by just typing *create partition primary*, as the *create partition* command without a *size=* parameter tells Diskpart to use all the remaining space on the hard disk. I can see the result of my work by typing *list partition*.

Now, we've got our partitions, but they still need drive letters and formatting. I'll show you how to do that next month.

Minasi, Mark. "Initializing Windows Disks with Dispart." *Windows IT Pro* March 2010: 11.

## Formatting and Resizing Partitions with Diskpart

If you ever needed to change a volume's size, this tool is for you



Mark Minasi

In last month's "Initializing Windows Disks with Diskpart" I showed you how Diskpart lets you view, select, create, and obtain detailed information about disk partitions. This month, we'll make those partitions useful by formatting them and giving them drive letters, then we'll see how to resize an existing partition to make it larger or smaller.

In last month's example, we added an empty 24GB drive to a Windows system and created a 10GB partition by typing `select disk 1` (which pointed Diskpart to the second physical hard disk) and `create partition primary size=10240` (Diskpart prefers size information in megabytes). To complete this disk's setup, we need to give it a drive letter with the `Assign` command, then format it with the `Format` command.

The `Assign` command is simple: After you focus Diskpart on a partition or volume, you can give that partition/volume a drive letter (or change the existing drive letter) by typing

```
assign [letter=<letter>]
```

To set this partition's drive letter to T, for example, you would use

```
assign letter=t
```

(If you don't specify a letter, Diskpart automatically assigns the next available letter to the partition.)

Now, you need to format the disk before you can use it. The syntax of Diskpart's `Format` command is a bit different from the syntax of the native `Format` command that Windows OSs have had since DOS 1.0. It has many options, but in most cases these options will do the trick:

```
format fs=<filesystemtype> [quick] [label=<label>] [unit=<clustersize>]
```

For example, you could format the partition quickly, allow `Format` to use the default cluster size, and label it "Data drive" by typing

```
Format fs=ntfs label="Dta drive" quick
```

That command gives you a working disk volume, but what if you want to change the volume's size? Since Windows Vista, Diskpart has been able to expand or shrink a partition/volume. Why shrink an existing volume? I've had to do it on a number of Vista and Windows Server 2008 systems because Windows' useful BitLocker drive-encryption tool lets you encrypt entire OS drives—but only if you have the foresight to leave 1.5GB of unused space on the disk where the OS resides. Because Vista/Server 2008 Setup is sadly BitLocker-unaware (a problem that Windows 7 and Server 2008 R2 don't share), many Vista/Server 2008 users carefully set up their systems, add BitLocker as a final touch, then find that BitLocker won't work without a 1.5GB partition. Oops!

I've used Diskpart to help many people in this situation. The tool's `Shrink` command reduces an existing partition's size without damaging that partition. To shrink a partition/volume, I'd first select that partition or volume. For example, if I

want to shrink the C drive on a system by 1.5GB, I would type *list volume* to determine the volume number that specifies the C drive (e.g., volume 2), select that volume by typing *select volume 2*, then type

```
shrink [desired=<sizeinmegabytes>] [querymax]
```

In my example, I need to clear 1,500MB of free space so that I can create the partition that will make BitLocker happy. If I just type *shrink* without any parameters, Diskpart computes the maximum space it can extract from C, then shrink C by that amount. But I don't want C minimized in size; I just want 1,500MB taken from it. So, I'll add the *desired=* parameter:

```
Shrink desire=1500
```

That command will give me the 1.5GB of space I need to set up that extra drive letter that Vista/Server 2008's BitLocker needs. To see how much space you can snatch from an existing drive, you can type *shrink querymax*.

Consider the opposite situation. You have a volume on a drive that doesn't fill that drive, leaving some precious disk space unused. How do you expand the volume to use all remaining space? You can use Diskpart's *Extend* command:

```
extend [size=<sizeinmegabytes>]
```

As with *Shrink*, first shift Diskpart's focus to the volume/partition you want to extend. Then, either type simply *extend*, which causes Diskpart to expand the volume/partition as much as possible, or constrain the extension with the *size=* parameter:

```
extend size=100
```

Diskpart's ability to expand and shrink volumes is a welcome addition to the list of built-in Windows storage tools.

Minasi, Mark. "Formatting and Resizing Partitions with Dispart." [Windows IT Pro](#) April 2010: 11