How Windows Starts Up (part 1 of 4)

ntdebug 19 Jun 2007 4:29 AM 11

Hi folks, my name is David and I'm an Escalation Engineer for Microsoft. Since Bryan wrote about **How Windows Shuts Down**, I thought it would be a good idea to cover How Windows Starts Up.

This information applies specifically to Windows 2000, Windows XP, and Windows Server 2003. I will blog separately on the boot changes in Windows Vista.

Additional information about this topic may be found in Chapter 5 of Microsoft Windows Internals by Russinovich and Solomon and also on TechNet:
http://technet.microsoft.com/en-us/library/bb457123.aspx

Methodology
The key to understanding and troubleshooting system startup issues is to accurately ascertain the point of failure. To facilitate this determination, I have divided the boot process into the following four phases.

1. Initial
2. Boot Loader
3. Kernel
4. Logon

Over the next few weeks, I'll be describing each of these phases in detail and providing appropriate guidance for troubleshooting relevant issues for each phase.

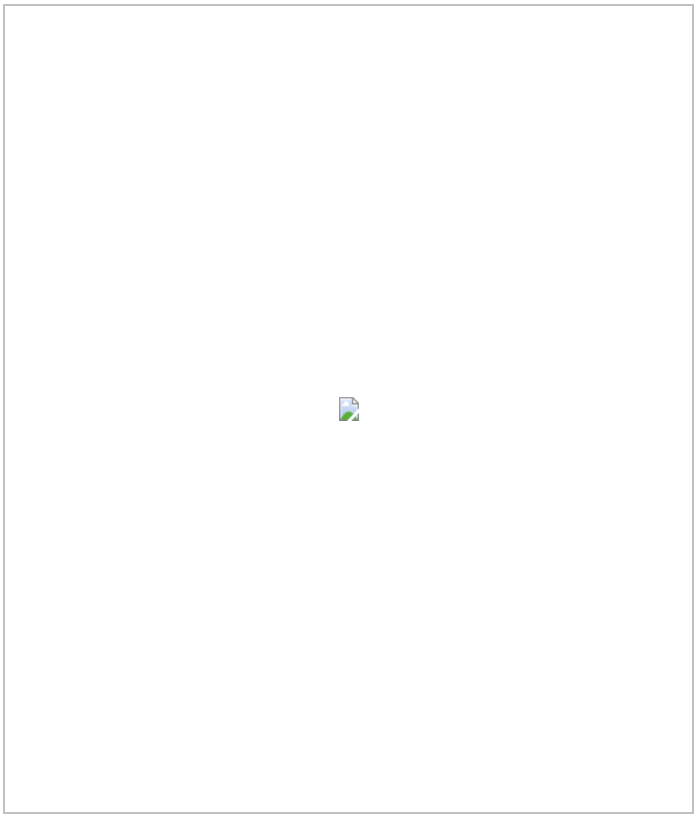
Initial Phase
The Initial Phase of boot is divided into the Power-On Self Test (POST) and Initial Disk Access.

Power-On Self Test
POST activities are fully implemented by the computer's BIOS and vary by manufacturer. Please refer to the technical documentation provided with your hardware for details. However, regardless of manufacturer, certain generic actions are performed to ensure stable voltage, check RAM, enable interrupts for system usage, initialize the video adapter, scan for peripheral cards and perform a memory test if necessary. Depending on manufacturer and configuration, a single beep usually indicates a successful POST.

Troubleshooting the POST
- ✓ Make sure you have the latest BIOS and firmware updates for the hardware installed in the system.
- ✓ Replace the CMOS battery if it has failed.
- ✓ Investigate recently added hardware (RAM, Video cards, SCSI adapters, etc.)
- ✓ Remove recently added RAM modules.
- ✓ Remove all adapter cards, then replace individually, ensuring they are properly seated.
- ✓ Move adapter cards to other slots on the motherboard.
- ✓ If the computer still will not complete POST, contact your manufacturer.

Initial Disk Access
Depending on the boot device order specified in your computer's BIOS, your computer may attempt to boot from a CD-ROM, Network card, Floppy disk, USB device or a hard disk. For the purposes of this document, we'll assume that we're booting to a hard disk since that is the most common scenario.

Before we discuss the sequence of events that occur during this phase of startup, we need to understand a little bit about the layout of the boot disk. The structure of the hard disk can be visualized this way: (Obviously, these data areas are not to scale)



Hard disks are divided into Cylinders, Heads and Sectors. A sector is the smallest physical storage unit on a disk and is almost always 512 bytes in size. For more information about the physical structure of a hard disk, please refer to the following Resource Kit chapter:
http://www.microsoft.com/resources/documentation/windowsnt/4/server/reskit/en-us/resguide/diskover.mspx

There are two disk sectors critical to starting the computer that we'll be discussing in detail:

- Master Boot Record (MBR)
- Boot Sector

The MBR is always located at Sector 1 of Cylinder 0, Head 0 of each physical disk. The Boot Sector resides at Sector 1 of each partition. These sectors contain both executable code and the data required to run the code.

Please note that there is some ambiguity involved in sector numbering. Cylinder/Head/Sector (CHS) notation begins numbering at C0/H0/S1. However, Absolute Sector numbering begins numbering at zero. Absolute Sector numbering is often used in disk editing utilities such as **DskProbe**. These differences are discussed in the following knowledge base article:

Q97819 Ambiguous References to Sector One and Sector Zero
http://support.microsoft.com/default.aspx?scid=kb;EN-US;Q97819

Now that we have that straight, when does this information get written to disk? The MBR is created when the disk is partitioned. The Boot Sector is created when you format a volume. The MBR contains a small amount of executable code called the Master Boot Code, the Disk Signature and the partition table for the disk. At the end of the MBR is a 2-byte structure called a Signature Word or End of Sector marker, which should always be set to 0x55AA. A Signature Word also marks the end of an Extended Boot Record (EBR) and the Boot Sector.

The Disk Signature, a unique number at offset 0x1B8, identifies the disk to the operating system. Windows 2000 and higher operating systems use the disk signature as an index to store and retrieve information about the disk in the registry subkey HKLM\System\MountedDevices.

The Partition Table is a 64-byte data structure within the MBR used to identify the type and location of partitions on a hard disk. Each partition table entry is 16 bytes long (four entries max). Each entry starts at a predetermined offset from the beginning of the sector as follows:

Partition 1	0x1BE	(446)
Partition 2	0x1CE	(462)
Partition 3	0x1DE	(478)
Partition 4	0x1EE	(494)

The following is a partial example of a sample MBR showing three partition table entries in-use and one empty:

```
000001B0:                                               80 01
000001C0: 01 00 07 FE BF 09 3F 00 - 00 00 4B F5 7F 00 00 00
000001D0: 81 0A 07 FE FF FF 8A F5 – 7F 00 3D 26 9C 00 00 00
000001E0: C1 FF 05 FE FF FF C7 1B – 1C 01 D6 96 92 00 00 00
000001F0: 00 00 00 00 00 00 00 00 – 00 00 00 00 00 00
```

Let's take a look at each of the fields of a partition table entry individually. For each of these explanations, I'll use the first partition table entry above and highlight the relevant section. Keep in mind that these values are little-endian.

Byte Offset	Field Length	Sample Value	Field Description
0x1BE	8 Bits	0x80	**Boot Indicator** 00=Do Not Use for Booting 80=Active partition (Use for Booting)

```
000001B0:                                                 80 01
000001C0: 01 00 07 FE BF 09 3F 00 - 00 00 4B F5 7F 00
```

0x1BF	8 Bits	0x01	**Starting Head**

```
000001B0:                                                 80 01
000001C0: 01 00 07 FE BF 09 3F 00 - 00 00 4B F5 7F 00
```

0x1C0 0x1C1	2 Byte Word	0x01	**Starting Sector** Only the first 6 bits are used. The upper 2 bits of this byte are used by the Starting Cylinder field.
		0x00	**Starting Cylinder** Uses 1 byte + 2 bits from the Starting Sector field to make up the cylinder value. The Starting Cylinder is a 10-bit number with a maximum value of 1023.

```
000001B0:                                                 80 01
000001C0: 01 00 07 FE BF 09 3F 00 - 00 00 4B F5 7F 00
```

0x1C2	8 Bits	0x07	**System ID** Defines the volume type. 0x07=NTFS

```
000001B0:                                                 80 01
000001C0: 01 00 07 FE BF 09 3F 00 - 00 00 4B F5 7F 00
```

Other Possible System ID Values:

Partition Type	ID Value
0x01	FAT12 primary partition or logical drive (fewer than 32,680 sectors in the volume)
0x04	FAT16 partition or logical drive (32,680–65,535 sectors or 16 MB–33 MB)
0x05	Extended partition
0x06	BIGDOS FAT16 partition or logical drive (33 MB–4 GB)
0x07	Installable File System (NTFS partition or logical drive)
0x0B	FAT32 partition or logical drive
0x0C	FAT32 partition or logical drive using BIOS INT 13h extensions
0x0E	BIGDOS FAT16 partition or logical drive using BIOS INT 13h extensions
0x0F	Extended partition using BIOS INT 13h extensions
0x12	EISA partition
0x42	Dynamic disk volume
0x86	Legacy FT FAT16 disk *
0x87	Legacy FT NTFS disk *
0x8B	Legacy FT volume formatted with FAT32 *
0x8C	Legacy FT volume using BIOS INT 13h extensions formatted with FAT32 *

Partition types denoted with an asterisk (*) indicate that they are also used to designate non-FT configurations such as striped and spanned volumes.

0x1C3	8 Bits	0xFE	**Ending Head** (0xFE=254 decimal)

```
000001B0:                                                 80 01
000001C0: 01 00 07 FE BF 09 3F 00 - 00 00 4B F5 7F 00
```

0x1C4 0x1C5	2 Byte Word	0xBF	**Ending Sector** As with the Starting Sector, it only uses the first 6 bits of the byte. The upper 2 bits are used by the Ending Cylinder field.
		0x09	**Ending Cylinder** Uses 1 byte in addition to the upper 2 bits from the Ending Sector field to make up the cylinder value. The Ending Cylinder is a 10-bit number with a maximum value of 1023.

```
000001B0:                                                 80 01
000001C0: 01 00 07 FE BF 09 3F 00 - 00 00 4B F5 7F 00
```

0x1C6	32 Bits	0x3F000000	**Relative Sectors** The offset from the beginning of the disk to the beginning of the volume, counting by sectors. 0x0000003F = 63

```
000001B0:                                                 80 01
000001C0: 01 00 07 FE BF 09 3F 00 - 00 00 4B F5 7F 00
```

0x1CA	32 Bits	0X4BF57F00	**Total Sectors** The total number of sectors in the volume. 0x007FF54B = 8,385,867 Sectors = 4GB

```
000001B0:                                                 80 01
000001C0: 01 00 07 FE BF 09 3F 00 - 00 00 4B F5 7F 00
```

Are you with me so far? Good! Now, Cylinder/Sector encoding can be a bit tricky, so let's take a closer look.

Cylinder - Sector Encoding															
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Cylinder bits 1-8								Cyl bits 9 & 10		Sector value Bits 1-6					

As you can see, the Sector value occupies the lower 6 bits of the word and the Cylinder occupies the upper 10 bits of the word. In our example, the starting values for Cylinder and Sector are 01 00. Values in the MBR use reverse-byte ordering, which is also referred to as 'little-endian' notation. Therefore, we swap the bytes and find that our starting values are Cyl 0, Sec 1.

Our ending values are more interesting: BF 09. First, we swap the bytes and obtain a hex value of 0x09BF. This value in binary notation is 100110111111. The following table illustrates how we derive the correct partition table values from these bytes:

Example: BF 09															
8	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**10**	**9**	**6**	**5**	**4**	**3**	**2**	**1**
0	0	0	0	1	0	0	1	1	0	1	1	1	1	1	1
10 Cylinder value bits 1-8								Cyl bits 9 & 10		Sector value bits 1-6					

The 6 low bits are all set to 1, therefore our Sector value is 111111 or 63. You can see above how the bits are arranged for the Cylinder value. The value above is 1000001001 (521). Since both Cylinder and Head values begin numbering at zero, we have a total of 522 Cylinders and 255 Heads represented here. This gives us an ending CHS value of: 522 x 255 x 63 = 8,385,930 sectors.

Subtracting the starting CHS address (Cylinder 0, Head 1, Sector 1) (63) gives us the total size of this partition: 8,385,867 sectors or 4GB. We can verify this number by comparing it to the Total Sectors represented in the partition table: 4B F5 7F 00. Applying reverse-byte ordering gives us 00 7F F5 4B which equals 8,385,867 sectors.

So, now that we have an understanding of what is contained within the structures on the disk, let's look at the sequence of events that occur. Remember, this is just after POST has successfully completed.

1. The motherboard ROM BIOS attempts to access the first boot device specified in the BIOS. (This is typically user configurable and can be edited using the BIOS configuration utility.)

2. The ROM BIOS reads Cylinder 0, Head 0, and Sector 1 of the first boot device.

3. The ROM BIOS loads that sector into memory and tests it.
   a. For a floppy drive, the first sector is a FAT Boot Sector.
   b. For a hard drive, the first sector is the Master Boot Record (MBR).

4. When booting to the hard drive, the ROM BIOS looks at the last two signature bytes of Sector 1 and verifies they are equal to 55AA.
   a. If the signature bytes do not equal 55AA the system assumes that the MBR is corrupt or the hard drive has never been partitioned. This invokes BIOS Interrupt 18, which displays an error that is BIOS-vendor specific such as **"Operating System not found"**.
   b. If the BIOS finds that the last two bytes are 55AA, the MBR program executes.

5. The MBR searches the partition table for a boot indicator byte 0x80 indicating an active partition.
   a. If no active partition is found, BIOS Interrupt 18 is invoked and a BIOS error is displayed such as **"Operating System not found"**.
   b. If any boot indicator in the MBR has a value other than 0x80 or 0x00, or if more than one boot indicator indicates an active partition (0x80), the system stops and displays **"Invalid partition table**".
   c. If an active partition is found, that partition's Boot Sector is loaded and tested.

6. The MBR loads the active partition's Boot Sector and tests it for 55AA.
   a. If the Boot Sector cannot be read after five retries, the system halts and displays "**Error loading operating system**".
   b. If the Boot Sector can be read but is missing its 55AA marker, "**Missing operating system**" is displayed and the system halts.
   c. The bootstrap area is made up of a total of 16 sectors (0-15). If sector 0 for the bootstrap code is valid, but there is corruption in sectors 1-15, you may get a black screen with no error. In this case, it may be possible to transplant sectors 1-15 (not Sector 0) from another NTFS partition using

DskProbe.

d. If the Boot Sector is loaded and it passes the 55AA test, the MBR passes control over to the active partition's Boot Sector.

7. The active partition's Boot Sector begins executing and looks for NTLDR. The contents of the Boot Sector are entirely dependent on the format of the partition. For example, if the boot partition is a FAT partition, Windows writes code to the Boot Sector that understands the FAT file system. However, if the partition is NTFS, Windows writes NTFS-capable code. The role of the Boot Sector code is to give Windows just enough information about the structure and format of a logical drive to enable it to read the NTLDR file from the root directory. The errors at this point of the boot process are file system specific. The following are possible errors with FAT and NTFS Boot Sectors.
   a. **FAT:** In all cases it displays "press any key to restart" after either of the following errors.
      (1) If NTLDR is not found "**NTLDR is missing**" is displayed
      (2) If NTLDR is on a bad sector, "**Disk Error**" displays.
   b. **NTFS:** In all cases it displays "Press CTRL+ALT+DEL to restart" after any of the following errors.
      (1) If NTLDR is on a bad sector, "**A disk read error occurred**" is displayed. This message may also be displayed on Windows 2000 or higher systems if Extended Int13 calls are required, but have been disabled in the CMOS or SCSI BIOS. This behavior may also be seen if an FRS (File Record Segment) cannot be loaded or if any NTFS Metadata information is corrupt.
      (2) If NTLDR is not found, "**NTLDR is missing**" displays.
      (3) If NTLDR is compressed, "**NTLDR is compressed**" displays.

8. Once NTLDR is found, it is loaded into memory and executed. At this point the Boot Loader phase begins.

Troubleshooting the Initial Phase (Initial Disk Access)
If you are unable to boot Windows and the failure is occurring very early in the boot process, begin by creating a Windows boot floppy and using it to attempt to boot the system.

> Q119467 How to Create a Bootable Disk for an NTFS or FAT Partition
> http://support.microsoft.com/default.aspx?scid=kb;EN-US;Q119467

If you are unable to boot to Windows NT with a floppy, skip to the section titled **"Unable to boot using floppy"**.

If you can successfully start the computer from the boot disk, the problem is limited to the **Master Boot Record**, the **Boot Sector**, or one or more of the following files: NTLDR, NTDetect.com, or Boot.ini. After Windows is running, you should **immediately** back up all data before you attempt to fix the Boot Sector or Master Boot Record.

1. **Run a current virus scanning program to verify that no virus is present.**
2. **Select the method of troubleshooting based upon the error generated.**
   - **Operating System not found**
     Typical BIOS-specific error indicating no 55AA signature on the MBR. This must be repaired manually using a disk editor such as DskProbe.
     Warning: *Running FixMBR or FDISK /MBR in this case will clear the partition table.*
   - **Invalid Partition Table**
     May be repaired using DiskProbe – check for multiple 0x80 boot indicators.
   - **Error Loading Operating System**
     This error indicates an invalid Boot Sector. Use the "Repairing the Boot Sector" section below.
   - **Missing Operating System**
     This error indicates that the 55AA signature is missing from the Boot Sector. This may be replaced using DskProbe or by using the "Repairing the Boot Sector" section below.
   - **NTLDR is missing**
     Use the "Repairing the NTLDR file" section below.
   - **Disk error (FAT only)**
     Indicates that NTLDR is located on a bad sector.
   - **A disk read error occurred (NTFS only)**
     Indicates that NTLDR is on a bad sector. Use the "Repairing the NTLDR file" section below.
   - **NTLDR is compressed**
     Indicates that NTLDR is compressed. Uncompress or replace NTLDR. Note: This error is rare.
   - **Computer boots to a black screen with blinking cursor**
     Could be an issue with the MBR or the active partition's boot code (**Q228734**). Use the appropriate section below.

Repairing the MBR
Use the FIXMBR command from the Windows Recovery Console to re-write the boot code located in the first 446 bytes of the MBR, but to leave the partition table intact.

> 326215 How To Use the Recovery Console on a Windows Server 2003-Based Computer That Does Not Start
> http://support.microsoft.com/default.aspx?scid=kb;EN-US;326215
>
> ***WARNING:*** *DO NOT use this method if you are receiving* ***"Operating System not found"*** *or other BIOS-specific error.* FIXMBR *will zero out the partition table if the 55AA signature is missing from the MBR. The data will be unrecoverable.*

Repairing the Boot Sector
Use the FIXBOOT command from the Windows Recovery Console to write a new Boot Sector to the system partition.

> 326215 How To Use the Recovery Console on a Windows Server 2003-Based Computer That Does Not Start
> http://support.microsoft.com/default.aspx?scid=kb;EN-US;326215

If you are not able to repair the Boot Sector by using this method, you may repair it manually using the following knowledge base article:

Q153973 Recovering NTFS Boot Sector on NTFS Partitions
http://support.microsoft.com/default.aspx?scid=kb;EN-US;Q153973

Repairing the NTLDR file
Use the Recovery Console to copy NTLDR from \i386 directory of the Windows CD-ROM to the root of the hard drive.

Unable to boot using floppy
If you are unable to start the computer with a boot floppy and you are not receiving an error message, then the problem is most likely with your partition tables. If invalid partitions are present and you are unable to start your computer with a boot disk, formatting the disk, reinstalling Windows and restoring from backup may be your only option. It may be possible to recreate the partition table using DskProbe if a backup of the partition information is available. It also may be possible to manually reconstruct the partition table however this is outside the scope of this blog post. We may cover this later.

If you do not have a current backup of the data on the computer, as a last resort a data recovery service may be able to recover some of your information.

Helpful knowledge base articles:

Q272395 Error Message: Boot Record Signature AA55 Not Found
http://support.microsoft.com/default.aspx?scid=kb;EN-US;Q272395

Q155892 Windows NT Boot Problem: Kernel File Is Missing From the Disk
http://support.microsoft.com/default.aspx?scid=kb;EN-US;Q155892

Q228004 Changing Active Partition Can Make Your System Unbootable
http://support.microsoft.com/default.aspx?scid=kb;EN-US;Q228004

Q155053 Black Screen on Boot
http://support.microsoft.com/default.aspx?scid=kb;EN-US;Q155053

Q314503 Computer Hangs with a Black Screen When You Start Windows
http://support.microsoft.com/default.aspx?scid=kb;EN-US;Q314503

Q153973 Recovering NTFS Boot Sector on NTFS Partitions
http://support.microsoft.com/default.aspx?scid=kb;EN-US;Q153973

Okay boys & girls, that's it for now. Next time, I'll continue with the boot sequence and cover the Boot Loader phase. This is where things really get interesting… ☺

Comments

CS44
20 Jun 2007 2:35 PM

Can't wait for the boot sector post! Thanks for the insight!

person
20 Jun 2007 5:27 PM

Note to self, never make an operating system.

Attila
23 Jun 2007 12:10 PM

Great stuff and great blog guys, keep up the good work.

10veer
24 Jun 2007 3:04 PM

Hay thats really great work.....

find it very useful

Abu
26 Jun 2007 10:26 AM

Really enjoy reading the articles in 'ntdebugging'. Eagerly awaiting more write ups.

Abu

Mike
26 Jun 2007 1:20 PM

woof... my XP PC won't boot windows (but will boot linux (dual boot)) after a power failure yesterday

how about getting part 2 (Boot Loader) up post haste?

George
16 Jul 2007 2:17 PM

Interesting, that young people usually find old stuff magic, even myself :) TEH booting was something boring a decade or two ago...

Dush
10 Sep 2008 5:04 AM

I was hoping to find something about booting from dynamic disks.

Neill
29 Jun 2009 11:46 AM

How Windows Starts Up (part 1 of 4)

where are parts 2,3 & 4 URLs please

<DIV class=commentowner>[Hello Neill. David wrote a part two which is located here -
http://blogs.msdn.com/ntdebugging/archive/2007/06/28/how-windows-starts-up-part-the-second.aspx

Last week I added a BING search control on the top right corner to make the Blog site more search friendly. Hope it helps:)

I'll ask him to get started on parts 3 and 4.

Thanks for reading the blog.
-ronsto]</DIV>

Shane
9 Jul 2009 5:34 PM

Very useful...I hope part 3 & 4 aren't too far away?

securitybay.co.uk
4 Aug 2010 1:44 AM

Informative and insightful. Thank you.