

# NTFS MASTER FILE TABLE

By Mark E. Donaldson

On an NTFS volume, all the data structures required to keep track of the NTFS volume are files. This is very different from a FAT volume, where, for example, the FAT is not a file, but some data put aside in a different area on the volume, not contained in or accessible as a file.

I'll give you an example. On an NTFS volume, the volume boot-record is a file. In fact it's a file called \$Boot, contained in the root directory of the NTFS volume. You can't see this file when you do a Command-Prompt DIR because its attributes don't allow it. But now that you know its name, you can go to a Command Prompt and type:

**ATTRIB \$Boot**

and sure enough, you'll get data back saying it's a SH (SYSTEM HIDDEN) file.

Try as you might, though, you can't use ATTRIB to change the attributes on it; that's NTFS's way of keeping you from doing something you'd regret. However, now that you know its name, you can type in

**DIR \$Boot /ASH**

and get a report like this:

```
D:\>dir $Boot /ASH
Volume in drive D is NTFSMAIN
Volume Serial Number is 6473-ADD1
```

Directory of D:\

```
07/17/98 05:31p          8,192 $Boot
           1 File(s)      8,192 bytes
           295,895,040 bytes free
```

The files containing the key data about an NTFS value are collectively called METADATA. The prefix META- means "above" or "higher". Thus the METADATA is data on the volume that "lives above" your data.

The key piece of metadata on an NTFS volume is the MFT -- the Master File Table. The MFT is named, by no coincidence, \$MFT. If you go to your command prompt and try the DIR command, you'll see it:

```
D:\>dir $MFT /ASH
Volume in drive D is NTFSMAIN
Volume Serial Number is 6473-ADD1
```

Directory of D:\

```
07/17/98 05:31p     15,250,432 $MFT
           1 File(s)  15,250,432 bytes
           295,895,040 bytes free
```

On this volume, the MFT is about 15MB.

# NTFS MASTER FILE TABLE

By Mark E. Donaldson

The MFT is internally divided into 1024-byte units called "MFT Records" or "File Record Segments" (FRSs). If you've jumped ahead of me, you'll have already grabbed your calculator and determined that  $15250432/1024=14893$ , which is exactly the number of MFT Records exist with the file called \$MFT on this volume.

The MFT Record for the MFT itself is at byte offset 0 in the MFT file. To access the MFT Record for the MFT, then, the operating system must read 1024 bytes out of the MFT file starting at offset 0. Hence, MFT record 0 (the first 1024 bytes of the MFT) describes the MFT. Therefore the MFT is also sometimes referred to as "file number 0".

For each file on an NTFS volume, there is at least one MFT record that describes it.

Each MFT record is 1024 bytes long. So if the operating system is looking for the \$Boot file (which is file #7) all the operating system does is multiply  $7*1024$  and start reading the MFT file itself at offset 7168 for 1024 bytes.

Reading random offsets into the MFT goes LOTS faster if the MFT is contiguous.

Now, if the cluster size on the volume is 512 bytes, the MFT records can actually be broken up into 2 pieces, thus requiring 2 I/Os to read the MFT record. This tends to be especially true on volumes that have been converted from FAT to NTFS.

So, the best thing to do is to have a contiguous MFT. Barring that, the best thing to do is to have a larger cluster size (minimally 1024, optimally 4096) so that you guarantee that the MFT records aren't split.

Now, the question may get asked, "Why do you keep insisting on 4096-byte clusters, when the problem of split MFT records is obviously solved by having a cluster size of 1024 bytes?"

Well, yes, the problem of split MFT records is solved by having a cluster size of 1024 bytes. But the problem of EXTENDING the MFT with minimal fragmentation is not. You see, the MFT is extended in chunks of 16 or 32 records. If the cluster size is 1024, then it may be that the MFT extension costs you 16 or 32 fragments. If the cluster size is 4096, then the MFT extension could only at most cost you 4 or 8 fragments. This is why I continue to harp on a 4096-byte cluster size.

The names of the basic metadata files (and their MFT record numbers) are:

- 0 - \$MFT
- 1 - \$MFTMirr
- 2 - \$LogFile
- 3 - \$Volume
- 4 - \$AttrDef
- 5 - . (which is the root directory),
- 6 - \$Bitmap
- 7 - \$Boot
- 8 - \$BadClus
- 9 - \$Quota (on NT4) \$Secure (on W2K)
- 10 - \$UpCase

# NTFS MASTER FILE TABLE

By Mark E. Donaldson

## 11 - \$Extend (on W2K)

There are metadata extensions in the \$Extend directory on Windows 2000, but they vary from system to system so I won't try to name them here.

**QUESTION:** 'With a 4096-byte cluster size, it's possible to fit 4 MFT records, with no splits, into a single cluster. This is much better for performance.' My question is... if you put 4 separate files in one cluster, how do you address them? One cluster... One address..."

**ANSWER:** I hope you can see from the above discussion that we aren't talking about putting 4 FILES into one cluster. We're only putting the MFT records DESCRIBING 4 files into one cluster, which is very different. Your assumption is correct: if there were 4 files, there'd be a problem fitting them into the same cluster, but we're only talking about 4K's worth of MFT records, which fit quite nicely into a 4K cluster.