NTFS File Attributes

By kexugit

Archived: 2026-04-05 18:26:14 UTC

▶
▶

It is time once again to delve into the fascinating world of NTFS! So far I've outlined how files become more complex as they grow, the different metafiles found in the MFT, and then the different parts of Windows storage that results in the infamous 2TB size limitation.

Today I want to list out most the different attribute types that a file can have. Think of these as the building blocks for the file itself. No file will have every attribute type. In fact most just have a few.

$STANDARD_INFORMATION – General information about the file. Creationtime, LastModificationTime, LastChangeTime, and LastAccessTime are all stored here.

NOTE: Even if the updates to LastAccessTime are disabled, the old time will still be stored here.

FileAttributes are also stored here. Do not confuse FileAttributes with the file's attribute types. FileAttributes are just flags and can mark the file as being...

Read-only	Archive
Hidden	Compressed
System	Encrypted

There are more flags but these are the really common ones.

So when you make changes to the file in this part of the UI, you are actually changing the FileAttributes flag of the $STANDARD_INFORMATION attribute type.

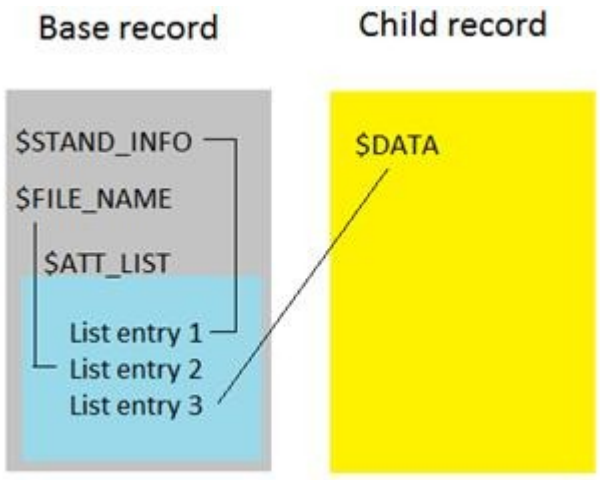
Also, this attribute holds a SecurityID. Do not confuse this with a security descriptor. Those are stored elsewhere. The SecurityID is used to help locate the correct security descriptor for the file in question.

$ATTRIBUTE_LIST – This attribute type keeps a list of all of the file's attribute types. But it only exists if at least ONE of the attribute types is nonresident. So if you have a file with five attribute types and one is nonresident, an $ATTRIBUTE_LIST will be added to the file.

To get a good visual of how the $ATTRIBUTE_LIST is used, refer to my blog on file growth.

https://blogs.technet.com/askcore/archive/2009/10/16/the-four-stages-of-ntfs-file-growth.aspx

But basically, the $ATTRIBUTE_LIST will tell you at what FRS (file record segment) you will find each of the file's attribute types. For attribute types that are resident, it will just point back to its own FRS.



$FILE_NAME - This is where we store the file name. No, really. In addition there are also fields for Creationtime, LastModificationTime, LastChangeTime, and LastAccessTime. These are not updated as often as their counterparts in the $STANDARD_INFORMATION attribute type.

This is also where we keep track of what directory the file belongs to. So if the parent directory incorrectly removes the still active file from its index, running CHKDSK will have enough information about where the file should live to be able to recover this 'orphaned file'.

$VOLUME_VERSION – This attribute type contains volume information, or at least it used to. It hasn't been used in a very long time. It only existed in early versions of Windows NT. I'm only including it in the list to be complete.

$OBJECT_ID – This is an attribute that holds an ID. This ID is used by the Distributed Link Tracking Service. An example of how it is used would be found in shortcuts. Make a shortcut on your desktop that points to a file. Then move that file. The shortcut will still function because it is using a way to tack the source file other than just the path and file name.

Not all files will have an $OBJECT_ID attribute. In fact, it isn't until an actual ID is to be assigned that the attribute is added to the file.

For more information about the Distributed Link Tracking Service please use the following MSDN link.

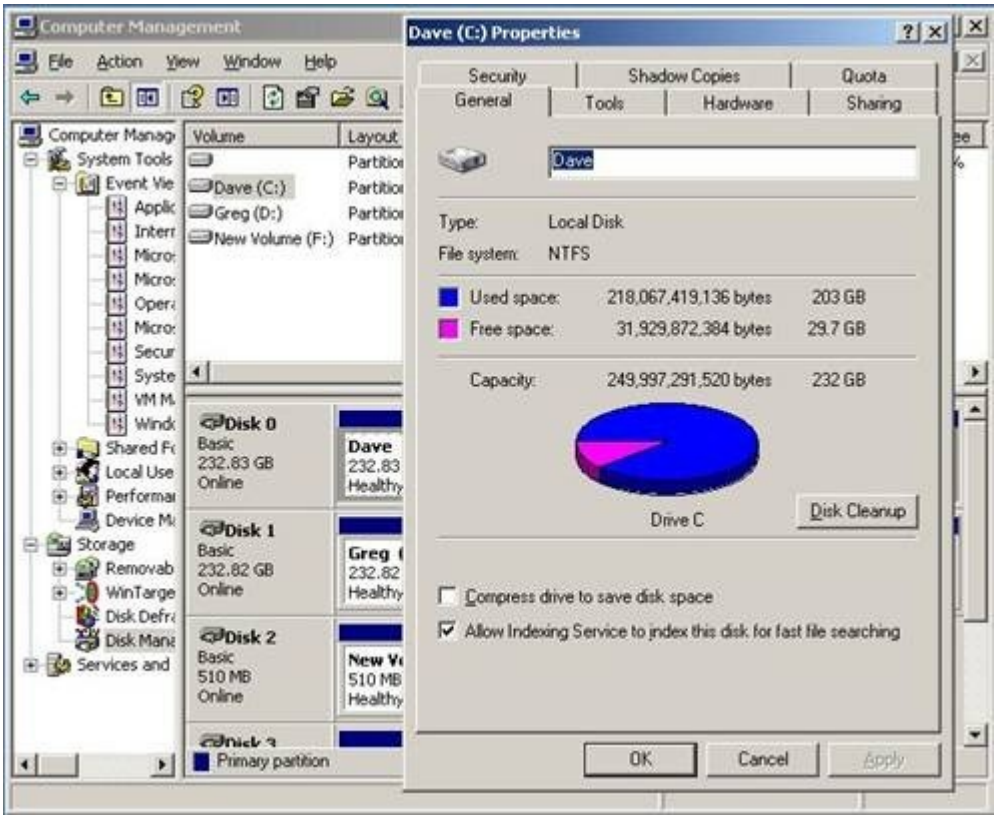
https://msdn.microsoft.com/en-us/library/aa363997%28VS.85%29.aspx

$SECURITY_DESCRIPTOR – This is where security information for the file used to be stored. In newer versions of NTFS, Microsoft moved to storing all security information in a single file called $SECURE. One of the main benefits of this was that files that had the same security on them didn't need to store that information in each individual file.

$VOLUME_NAME – This attribute exists in only one file in each NTFS volume...the $Volume file. This is one of the metafiles I listed in my blog NTFS Metafiles.

https://blogs.technet.com/askcore/archive/2009/12/30/ntfs-metafiles.aspx

It is in this file, in this attribute that the name or 'label' of the file is found.



My volume is here is called Dave. The default name of a volume is New Volume.

$VOLUME_INFORMATION – Also only found in the $Volume metafile, this attribute contains version information for NTFS and a field for volume flags such at the 'dirty bit'.

$DATA – When we think of a file, we typically just think of the data that is in it. The $DATA attribute is where that data is located. This attribute is normally only found in files...not directories.

$INDEX_ROOT – A directory will have an index that contains information about the files associated with that directory. If there are only a few index entries, they will all be found in the $INDEX_ROOT attribute. Once there are too many entries, they are moved to an $INDEX_ALLOCATION attribute. These index entries form a 'b-tree'.

The layout of indexes is somewhat complex and I will probably do a separate blog for them. For this list, I'm trying to keep it simple.

$INDEX_ALLOCATION – See $INDEX_ROOT

$BITMAP – The $BITMAP attribute is also part of the index structure. It keeps track of what parts of the index are allocated and which are free to be reused. It also provides a similar function to the $MFT file.

Do not confuse this attribute with the $BITMAP metafile.

$SYMBOLIC_LINK – Symbolic links, reparse points and hard links are commonly misunderstood. The $SYMBOLIC_LINK attribute is currently not in use. The MKLINK utility allows you to create symbolic links but it does so by creating a $REPARSE_POINT attribute.

$REPARSE_POINT – This attribute is used when a symbolic link or mount point is created. There will be a ReparseTag in the attribute that tells us what type of reparse point is being used.

There are other attribute types but for the most part they are obscure or obsolete. This list covers the ones people are likely to care about.

Robert Mitchell
Senior Support Escalation Engineer
Microsoft Enterprise Platforms Support
“Demystifying NTFS as much as I’m allowed”

- **Anonymous**
  January 01, 2003
  Sorry to leave this comment so long without an answer. I can see what you are trying to do, but it would be better accomplished by creating an alternate data stream (ADS) and storing your information there. It wouldn't affect the primary data stream (unnamed data stream) and it would follow the file. An ADS would actually exist in a separate data attribute.
- **Anonymous**
  September 28, 2011
  Robert, this is a good post!How extensible are NTFS attributes? Hypothetical example; To give developers a way of marking files and folders as migrateable (defined as non-machine specific, non-cache, non-temp user data), could a new entry be added to $STANDARD_INFORMATION to allow for this? Would this be a huge undertaking re compatibility and other issues, or fairly simple to implement? I'm thinking in terms of backup tools like Robocopy.
- **Anonymous**
  October 31, 2013
  It says under the $BITMAP attribute section "Do not confuse this attribute with the $BITMAP metafile". It should actually read "Do not confuse this attribute with the $Bitmap metafile" since metafile are given as lowercase to distinguish them from attributes! It would avoid confusion if everyone stuck to this convention.
- **Anonymous**
  October 18, 2014
  I am having problems with 3 USB external drives. My .jpg files are changed into folders and it is impossible to open o delete them.
- **Anonymous**
  December 19, 2015
  i use windows 10 but i am facing nfts local disk c (access denied) problem

---

Source: https://blogs.technet.microsoft.com/askcore/2010/08/25/ntfs-file-attributes/