

TZWorks® NTFS Copy Utility (*ntfscopy*) Users Guide



Abstract

ntfscopy is a standalone, command-line tool that can copy any file on a NTFS volume. It can operate on a *live* NTFS volume, an image of an NTFS volume or a *VMWare* volume. ***ntfscopy*** runs on Windows, Linux and macOS.

Copyright © TZWorks LLC

www.tzworks.com

Contact Info: info@tzworks.com

Document applies to v1.14 of ***ntfscopy***

Updated: Apr 25, 2025

Table of Contents

1	Introduction	2
2	How to Use <i>ntfscopy</i>	3
2.1	Copying a File from a Live Volume.....	4
2.2	Copying a File from an off-line Image	5
2.3	Copying a File from a <i>VMWare</i> Volume.....	6
2.4	Including Slack space in the Copy	6
2.5	Piping in filenames to copy	7
2.5.1	Using the built in dir command.....	7
2.5.2	Filtering out specific types of files	7
2.5.3	Using a list of files	8
2.5.4	Using Other Tools.....	8
2.6	Ignoring sparse clusters	9
2.7	Copying the NTFS Attributes.....	9
2.8	Copying Alternate Data Streams (ADS).....	10
2.9	Copying Files from Volume Shadows.....	12
3	Known Issues.....	13
3.1	Linux and macOS Specific.....	13
3.2	Windows Specific (Use of backslashes and quotes)	13
4	Available Options	13
5	Authentication and the License File.....	15
5.1	<i>Limited</i> versus <i>Demo</i> versus <i>Full</i> in the tool's Output Banner	15
6	References	15

TZWorks® NTFS Copy Utility (*ntfscopy*)

Users Guide

Copyright © TZWorks LLC

Webpage: http://www.tzworks.com/prototype_page.php?proto_id=9

Contact Information: info@tzworks.com

1 Introduction

ntfscopy is a prototype tool that can copy any file (or alternate data stream) from a NTFS file system. This can be from either from a *live* system or from an imaged NTFS volume or drive. The term *copy* as it is used here means it can be done in one of two modes: (a) 'raw' or (b) 'normal', where the latter is the default.

In '*raw*' mode, *ntfscopy* will copy all the data clusters allocated to the file in question. Consequently, this allows one to view the file contents as they were stored on the physical disk, including any slack space and any compression used.

The *default* mode is to copy only the clusters or portions of clusters, used to store valid data. So in this mode, no *slack* space will be copied. In addition, if the NTFS file incorporates the standard Windows disk compression, the *default* mode will try to uncompress the data and copy the data as it would be viewed if opening the file normally. The compression here is assumed to only be the native NTFS file compression as opposed to a third party compression scheme.

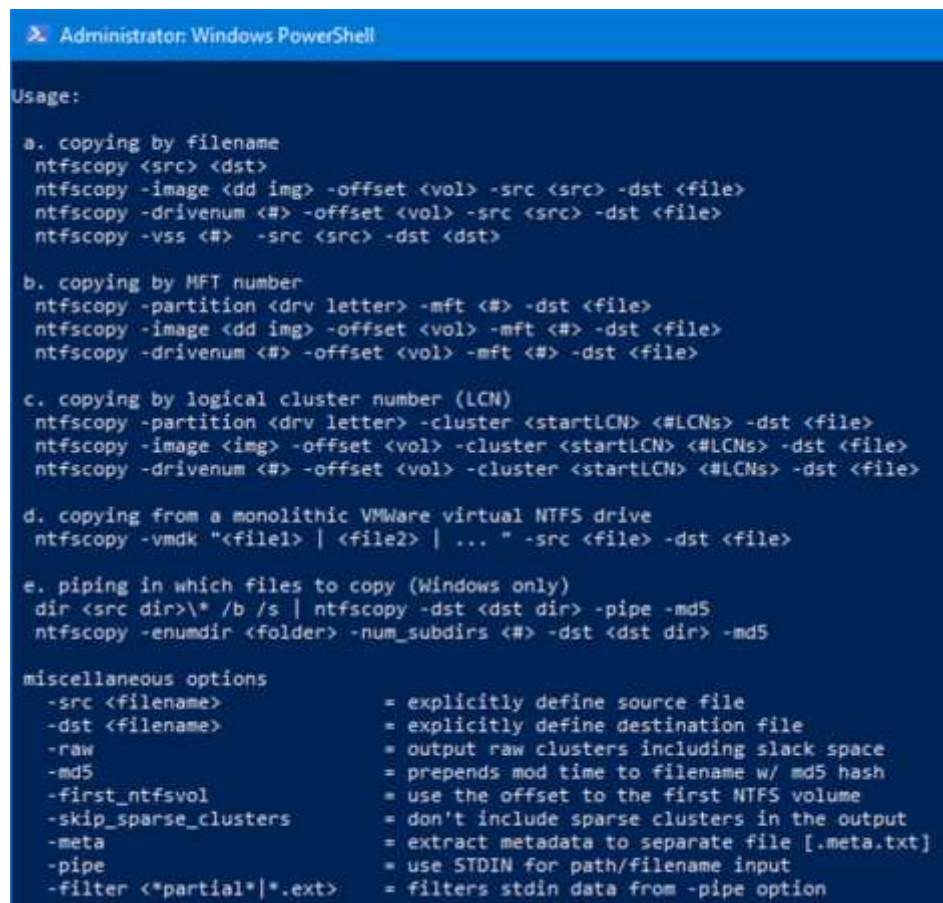
ntfscopy is able to copy any file on the file system since it accesses the hard drive at the cluster level. In its current form, it does not rely on any unique drivers but handles the cluster to NTFS translation in the *usermode* space using its own parsing engine. What this means is it does not make use of any Windows NTFS unique API calls or custom device drivers. Reading at the cluster level allows ***ntfscopy*** to bypass the read permissions of the file and read any file with just administrative privileges. This includes files locked down by the operating system such as registry hives and other files open for exclusive access. The administrative privileges are needed only so ***ntfscopy*** can perform cluster reads. Using this approach, one maximizes the chances that the data that is being copied is not being transformed by some mechanism in-between your request to perform the copy and the actual copy operation. This becomes more of an issue if there is some malware or root-kit on the Windows machine you are investigating.

ntfscopy has been compiled to run on Windows, Linux or Mac's OSX. From the Windows perspective, it should work on Windows XP up to Windows 10.

2 How to Use *ntfscopy*

ntfscopy requires administrative privileges to copy from a *live* Windows system to obtain read access to the clusters on the drive; otherwise it will be restricted to copying files from off-line images. Therefore to perform *live* processing of volumes, one needs to launch the command prompt with administrator privileges.

One can display the menu options by typing in the executable name without parameters. A screen shot of the menu is shown below. From the available options, one can copy files using a number of options: (a) from a live NTFS volume, (b) from a captured image with an NTFS partition, or (c) from a *VMWare* volume.

A screenshot of a Windows PowerShell window titled 'Administrator: Windows PowerShell'. The window displays the usage instructions for the 'ntfscopy' command. The text is as follows:

```
Usage:

a. copying by filename
ntfscopy <src> <dst>
ntfscopy -image <dd img> -offset <vol> -src <src> -dst <file>
ntfscopy -drivenum <#> -offset <vol> -src <src> -dst <file>
ntfscopy -vss <#> -src <src> -dst <dst>

b. copying by MFT number
ntfscopy -partition <drv letter> -mft <#> -dst <file>
ntfscopy -image <dd img> -offset <vol> -mft <#> -dst <file>
ntfscopy -drivenum <#> -offset <vol> -mft <#> -dst <file>

c. copying by logical cluster number (LCN)
ntfscopy -partition <drv letter> -cluster <startLCN> <#LCNs> -dst <file>
ntfscopy -image <img> -offset <vol> -cluster <startLCN> <#LCNs> -dst <file>
ntfscopy -drivenum <#> -offset <vol> -cluster <startLCN> <#LCNs> -dst <file>

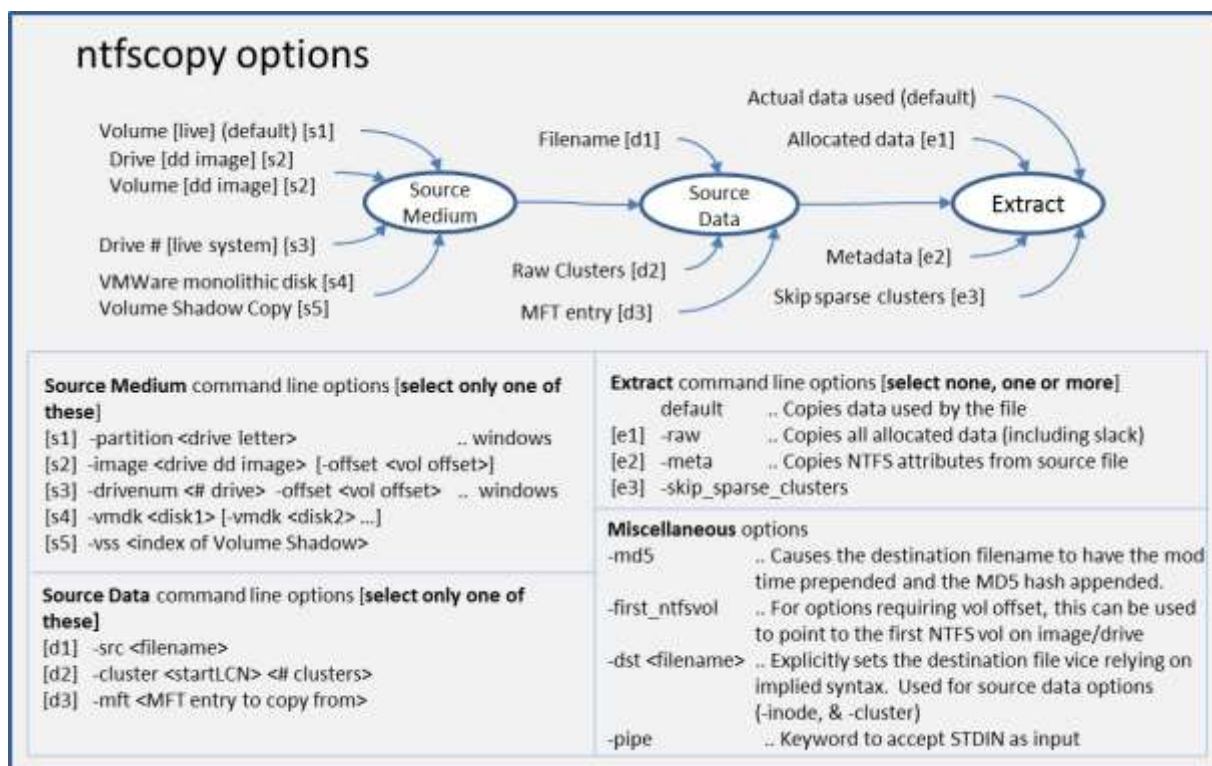
d. copying from a monolithic VMWare virtual NTFS drive
ntfscopy -vmdk "<file1> | <file2> | ... " -src <file> -dst <file>

e. piping in which files to copy (Windows only)
dir <src dir> \* /b /s | ntfscopy -dst <dst dir> -pipe -md5
ntfscopy -enumdir <folder> -num_subdirs <#> -dst <dst dir> -md5

miscellaneous options
-src <filename>           = explicitly define source file
-dst <filename>           = explicitly define destination file
-raw                     = output raw clusters including slack space
-md5                     = prepends mod time to filename w/ md5 hash
-first_ntfsvol           = use the offset to the first NTFS volume
-skip_sparse_clusters    = don't include sparse clusters in the output
-meta                    = extract metadata to separate file [.meta.txt]
-pipe                    = use STDIN for path/filename input
-filter <*<partial*>|.ext> = filters stdin data from -pipe option
```

After selecting a source medium to copy from, one selects the target file. This could include: (a) a path/filename, (b) the MFT entry of the file, or (c) the '*cluster run*' one is interested in extracting. There are also optional parameters, so one can select various switches to cause the copy to behave in different ways.

The figure below shows the possible choices and where they apply in a logical processing flow. The comments in the figure annotate any restrictions for a particular option.



2.1 Copying a File from a Live Volume

To copy a file from a live NTFS volume (or partition), one has two choices: (a) specify the absolute path of the source file, or (b) specify the drive number and volume offset by using the **-drivenum <num> -offset <volume offset>** option. Either choice accomplishes the same task. The first choice is more straightforward and easier to use. The second choice is more complex, but allows one to target hidden NTFS partitions that do not have a drive letter. A couple examples are shown below:

```
ntfscopy c:\$boot c:\dump\vol_boot.bin
```

```
ntfscopy -drivenum 0 -offset 0x100000 -src \$boot -dst c:\dump\vol_boot.bin
```

```
ntfscopy -drivenum 0 -first_ntfsvol src \$boot -dst c:\dump\vol_boot.bin
```

The first example targets the hidden `$boot` file in the 'c' partition and copies the contents to the `c:\dump\vol_boot.bin` file. The second example copies the `$boot` file from the volume that starts at offset `0x100000` hex relative to drive 0. In this case, offset of `0x100000` is the first NTFS partition on my drive 0. It also happens to be a hidden partition. Included in the syntax is the **-src <file>** and **-dst <file>** to explicitly define the source and destination files. The third example yields the same result as the second, but allows one to target the first NTFS partition without specifying any offsets via the **-first_ntfsvol** option. This is useful when scripting **ntfscopy** to pull files from any box without worrying what is the offset of the first NTFS volume is.

Another example of pulling data from a *live* volume is to use the **-cluster <starting Logical Cluster Number> <# clusters>** option. Geared more for the reverser, this allows one to pull any *cluster run* from a volume. An example is shown below:

```
ntfscopy -partition c -cluster 0 4 -dst c:\dump\cluster_dump.bin
```

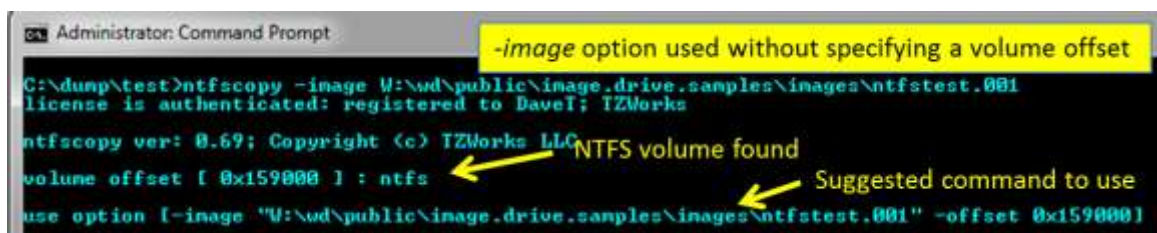
The above example pulls the first 4 clusters from the 'c' partition. Instead of using the source medium of **-partition <drive letter>**, one could have specified it to be **-drivenum <num> -offset <vol offset>** as was used in the examples earlier.

2.2 Copying a File from an off-line Image

To process an image that has been already acquired, and is in the 'dd' format, one uses the **-image** switch. This option can be used in two flavors. If the image is of an entire drive, then one needs to explicitly specify the *offset* of the location of the volume you wish to target. On the other hand, if the image is only of a volume, then you do not need to specify the offset of the volume (since it is presumed to be at offset 0).

For the first case, where an offset needs to be explicitly specified, **ntfscopy** will help the user in locating where the NTFS volume offsets are. If one just issues the **-image** command without the offset, and there is *not* a NTFS volume at offset 0 (eg. second case mentioned above), **ntfscopy** will proceed to look at the master boot record contained in the image, determine where the NTFS partitions are, and report them to the user. This behavior was meant to be an aid to the user so that one does not need to resort to other tools to determine where the offsets for the NTFS volumes are in an image. Below is a screenshot what is displayed to the user in this situation.

Shown in the screenshot is a **-image** command that is issued without the offset. **ntfscopy** detects that the image is of an entire drive versus of a volume and locates one NTFS volume at offset 0x159000 hex. **ntfscopy** then reports to the user a suggested syntax (of the command, if re-issued) to process this volume.



```
Administrator: Command Prompt
C:\dump\test>ntfscopy -image W:\ud\public\image.drive.samples\images\ntfstest.001
license is authenticated: registered to DaveI; TZWorks
ntfscopy ver: 0.69; Copyright (c) TZWorks LLC
volume offset [ 0x159000 ] : ntfs
use option [-image "W:\ud\public\image.drive.samples\images\ntfstest.001" -offset 0x159000]
```

Another nuance with using images as the source is when specifying a path to a directory within the image to analyze. Since the image is not mounted as a drive, one really should not associate it with a drive letter when specifying the path. If one does do this, **ntfscopy** will ignore the drive letter and proceed to try to find the path starting at the root directory which is at MFT entry 5 for NTFS volumes.

In the below example, we copy the \$MFT record to a temporary directory to analyze it with some other tools.

```
ntfscopy -image c:\dump\my_image.dd -first_ntfsvol -src \MFT -dst c:\dump\vol_mft.bin
```

The **-src** and **-dst** keywords are used to identify the source and destination files. Note that the source does not have a drive letter, but the destination does. Finally, since we want to target the first NTFS volume, we use the shortcut switch of **-first_ntfsvol**. One could have specified the offset of the NTFS volume using the **-offset** option, as well.

2.3 Copying a File from a VMWare Volume

This option is still considered *experimental* since it has only been tested on a handful of configurations. Furthermore, this option is limited to *monolithic* type VMWare images versus *split* images. In VMWare, the term *split* image means the volume is separated into multiple files, while the term *monolithic* virtual disk is defined to be a virtual disk where everything is kept in one file. There may be more than one VMDK file in a *monolithic* architecture, where each monolithic VMDK file would represent a separate snapshot.

When working with virtual machines, the capability to handle snapshot images is important. Thus, if processing a VMWare snapshot, one needs to include the desired snapshot/image as well as its inheritance chain.

To handle an inheritance chain, **ntfscopy** can handle multiple VMDK files by using multiple **-vmdk <file>** switches, where each one represents a segment in the inheritance chain of VMDK files (eg. **-vmdk <VMWare NTFS virtual disk-1> ... -vmdk <VMWare NTFS virtual disk-x>**).

Aside from the VMDK inheritance chain, everything else is the same when using this option to that of normal 'dd' type images discussed in the previous section.

2.4 Including Slack space in the Copy

Each file that reserves allocated space will most likely not use all the allocated space. This allocated, but unused, space is called slack space. The contents of the slack space may contain some data from the previous owner of the cluster.

The default behavior of **ntfscopy** is to only copy the *valid* (or used) space of the file and discard any *slack* space. One can change this default behavior by invoking the **-raw** switch to copy *all* the allocated space associated with a file. This will then include all the unused space (or slack space) in the resulting file.

Another fine point with the default behavior is that any file that is stored using the native NTFS compression attribute will be uncompressed during the copy operation. However, with the **-raw** option, the copy operation will include a bit for bit copy of the allocated clusters. This means if the file is using the native NTFS compression, then the copied file will contain that structure as well.

2.5 Piping in filenames to copy

While copying one file at a time is useful, one may want to copy all the files in a directory or a set of sub-directories. One way to do this is to pipe in all the paths/filenames of the files one wishes to copy into **ntfscopy**. To allow **ntfscopy** to receive data from an input pipe, one needs to invoke the **-pipe** switch. This will put **ntfscopy** into a mode to receive a separate path/filename per line as input. The precondition when using the **-pipe** command is that the filename needs to include the absolute path as part of the input.

If one cannot use the **-pipe** option, one can use the experimental **-enumdir** option, which has similar functionality with more control. The **-enumdir** option takes as its parameter the folder to start with. It also allows one to specify the number of subdirectories to evaluate using the **-num_subdirs <#>** sub-option.

2.5.1 Using the built in dir command

One can use the Windows built-in **dir** command along with some of its internal switches to provide the input **ntfscopy** needs to copy files. Below is an example of a way to copy all the user hives from a *live* system into a separate directory:

```
dir c:\users\*ntuser.dat /b /s | ntfscopy -pipe -dst c:\dump\hives -md5
```

The command above makes use of the operating system's **dir** shell command. By specifying the **c:\users*ntuser.dat** as the criteria to start and filter during the search, the **dir** command will only return those files that contain the '*ntuser.dat*' character sequence. Using the switches **/b** and **/s**, the **dir** command will traverse all the subdirectories starting with the **c:\users** directory and return an absolute path to the files found without any extraneous information. The output returned from **dir** is '*piped*' into the **ntfscopy** application, which uses the **-pipe** switch, **-dst** switch, and the **-md5** switch. The **-pipe** switch tells **ntfscopy** to expect input from standard input. The **-dst** switch in combination with the **-pipe** switch says to put all the files copied into the directory specified by **-dst**, which in this case is **c:\dump\hives**. Finally, to ensure each *ntuser.dat* file copied has a unique name, to avoid being overwritten by another same named one, the **-md5** switch is used to append the md5 hash to the copied file.

2.5.2 Filtering out specific types of files

One can filter certain files to copy by using the **-filter** option. With this option, one can filter multiple partial names and/or extensions. This option allows the use of the star character '*' before and/or after a partial name to indicate the preceding or ending portion, respectively can be any set of characters. For example, if one wanted to copy all the Microsoft Office type files, one could specify the following: **-filter "*.pptx|*.docx|*.xlsx"**, to say you only want these types of files copied.

2.5.3 Using a list of files

One does not need to use the **dir** shell command, but could use a listing of filenames with their absolute paths and just echo out the list to the console while piping it into **ntfscopy** as shown above to achieve a similar result of copying many files in an automated fashion. For example, say one has a file that contains a list of files with absolute paths. One can print out the list using the Windows echo shell command like so:

```
echo filelisting.txt | ntfscopy -pipe -dst c:\dump\hives -md5
```

2.5.4 Using Other Tools

If you have access to our **vsenum** tool, then one can use it to send filenames to **ntfscopy**. The **vsenum** tool, while geared towards Volume Shadow Copies also has the functionality of just producing a directory listing of any folder on a Windows box. The other advantage of using **vsenum** is it has some additional options to allow it to filter which files are outputted as well as how deep to go down a directory tree. Below are some examples of use-cases:

- To copy all the user registry hives (ntuser.dat and usrclass.dat) to a temp directory

```
vsenum -dir c:\users -filter "ntuser.dat|usrclass.dat" -level 10 | ntfscopy -pipe -dst my_temp -md5
```

- To copy the critical OS hives to a temp directory

```
vsenum -dir %systemroot%\system32\config -filter "software|system|security|sam" |  
ntfscopy -pipe -dst my_temp -md5
```

If you have access to our **ntfsdir** tool, then one can use it to send filenames to **ntfscopy**. When using **ntfsdir**, one would need to use that switch **-filepath_only** to tell **ntfsdir** to only output the full path and filename without any metadata information, such as timestamp, size or MFT entry number. This is useful is enumerating various hidden files (eg. \$MFT, \$Boot, etc) and/or alternate data streams.

2.6 Ignoring sparse clusters

Windows uses the concept of sparse clusters for *reserving* a size for a file without actually using clusters. Normally when copying a file, one could care less about the sparse clusters. This is because sparse data is not backed by real clusters, and thus the sparse data is realized as just zeros

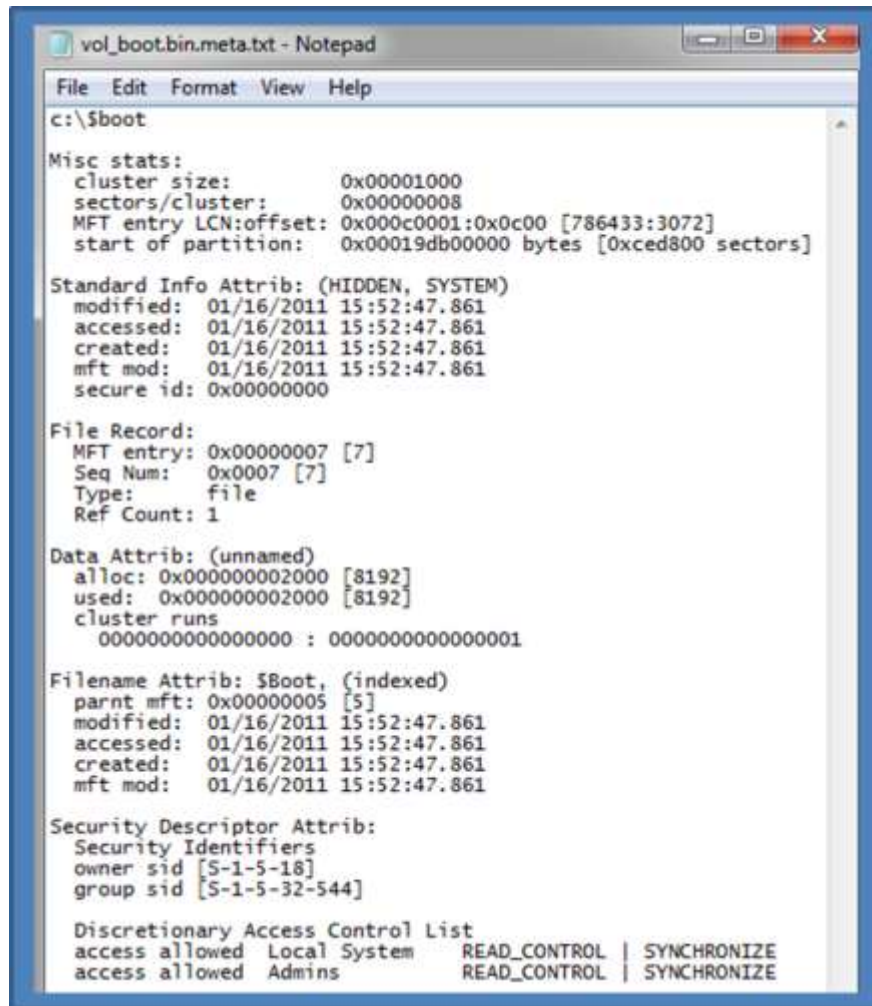
ntfscopy allows one to skip these types of clusters via the **-skip_sparse_clusters** switch. This option is important since blindly copying a file, that has many sparse clusters, may be much more data than you expected. A useful file in forensics that contains sparse data is the *change log journal*. The size of the change log journal can be significantly smaller when using this option. See the discussion on Alternate Data Streams for an example of using this option.

2.7 Copying the NTFS Attributes

If one wishes to copy the NTFS attributes associated with a file during the copy operation, one can invoke the **-meta** switch. This option tells **ntfscopy** to copy both the metadata and normal data associated with the file. Two files will be generated during this operation: (a) the normal copied file and (b) a separate file containing all the NTFS metadata. The name of the metadata file will be the same as the normal data file with the additional characters *“.meta.txt”* as a suffix. Below is an example that is taken from one of the sections above but with the **-meta** switch added:

```
ntfscopy c:\$boot c:\dump\vol_boot.bin -meta
```

The above results in 2 files: (a) *vol_boot.bin*, which contains the contents of *\$boot* and (b) the metadata file *vol_boot.bin.meta.txt*. Below is snapshot of the contents one should expect from the metadata file.



```
vol_boot.bin.meta.txt - Notepad
File Edit Format View Help
c:\$boot
Misc stats:
  cluster size:      0x00001000
  sectors/cluster:   0x00000008
  MFT entry LCN:offset: 0x000c0001:0x0c00 [786433:3072]
  start of partition: 0x00019db00000 bytes [0xcd800 sectors]
Standard Info Attrib: (HIDDEN, SYSTEM)
  modified: 01/16/2011 15:52:47.861
  accessed: 01/16/2011 15:52:47.861
  created: 01/16/2011 15:52:47.861
  mft mod: 01/16/2011 15:52:47.861
  secure id: 0x00000000
File Record:
  MFT entry: 0x00000007 [7]
  Seq Num: 0x0007 [7]
  Type: file
  Ref Count: 1
Data Attrib: (unnamed)
  alloc: 0x000000002000 [8192]
  used: 0x000000002000 [8192]
  cluster runs
    0000000000000000 : 0000000000000001
Filename Attrib: $Boot, (indexed)
  parnt mft: 0x00000005 [5]
  modified: 01/16/2011 15:52:47.861
  accessed: 01/16/2011 15:52:47.861
  created: 01/16/2011 15:52:47.861
  mft mod: 01/16/2011 15:52:47.861
Security Descriptor Attrib:
  Security Identifiers
  owner sid [S-1-5-18]
  group sid [S-1-5-32-544]
Discretionary Access Control List
  access allowed Local System READ_CONTROL | SYNCHRONIZE
  access allowed Admins READ_CONTROL | SYNCHRONIZE
```

2.8 Copying Alternate Data Streams (ADS)

In NTFS, each file consists of one or more data streams. Most files, however, contain one data stream and it is an *'unnamed stream'* which the operating system uses as the default (or primary) stream. Additional data streams, if present, will be *'named'* and are called Alternate Data Streams (ADS). These alternate streams provide a way for Windows to store additional data (metadata, security settings, or just a blob of data) within one file and allow each of the data entities it to be completely separate from the primary data or other ADS's. The ADS separation between each other includes not only a unique name, but also separate *cluster runs* to store its data if it is too large to fit within the NTFS attribute for the alternate stream.

The notation to address an alternate stream is to use the colon character as follows:

<filename>:<ADS>. Alternate streams are not displayed in Windows Explorer, and their size is not included in the file size that is reported to the user.

As an example, below is an important file in computer forensics, the NTFS *Change Log Journal*. As background, the Change Log Journal resides in the `[root]\$Extend\$UsnJrnl` file in the `$J` alternate data stream. On in Windows 7, the *UsnJrnl* collection is on by default for the primary partition, logging the all MFT entry changes on that volume.

Internals of the Change Log Journal Data Streams

```

----- attribute num 3 (relative offset 0x0108) -----
(0x00) type: 0x00000080, Data Attrib
(0x04) size: 0x00000060
(0x08) non_resident_flag: 0x01 (non_resident)
(0x0c) flags: 0x00
(0x0e) attrib_id: 0x0048
(0x10) starting_vcn: 0x8000 (sparse)
(0x14) ending_vcn: 0x0003
(0x18) run_array_offset: 0x0003
(0x1c) compression_flag: 0x04 (compressed)
(0x20) unk (skip 5 bytes): 00 00 00 00 00
(0x24) size_data_allocated: 0x00000000174f0000
(0x28) size_data_real: 0x00000000174b17f8
(0x2c) size_valid_data: 0x00000000174b17f8
(0x30) size_compressed: 0x0000000002120000
(0x48) name $J <-- ALTERNATE DATA STREAM
(0x50) data run
run: 0x00 [ 0x000050 ], [ 03 c0 53 01 ]
type_data: 0x80 (Data Attrib)
vcn (start): 0
vcn (end): 153bf
lcn (offset): 0
lcn (start): 0 (0)
lcn (stop): 153bf (86975)
num_clusters: 0x153c0
[above are sparse clusters] - not included in total below
run: 0x01 [ 0x000054 ], [ 32 30 21 c7 e8 37 ]
type_data: 0x80 (Data Attrib)
vcn (start): 153c0
vcn (end): 174ef
lcn (offset): 37e8c7
lcn (start): 37e8c7 (3664071)
lcn (stop): 3809f6 (3672566)
num_clusters: 0x2130
total num clusters: 0x2130

----- attribute num 4 (relative offset 0x0168) -----
(0x00) type: 0x00000080, Data Attrib
(0x04) size: 0x00000040
(0x08) non_resident_flag: 0x00 (resident)
(0x0c) flags: 0x04
(0x0e) offset_name: 0x0018
(0x10) offset_data: 0x0000
(0x14) attrib_id: 0x0005
(0x18) size_data: 0x00000020
(0x1c) offset_data: 0x0020
(0x20) resident_flags: 0x00
(0x24) resident_data: 0x00
(0x18) name $Max <-- ALTERNATE DATA STREAM
  
```

Sparse (reserved clusters)

Allocated clusters

Resident data

create: 01/10/10 06:05:39.187
 mod: 01/10/10 06:05:39.187
 access: 01/10/10 06:05:39.187
 meta: 01/10/10 06:05:39.187

Using our custom-built NTFS viewer, we took a snapshot of the *UsnJrnl* file. Highlighted are two named data streams: (a) the `$J` stream and (b) the `$Max` stream. The `$J` stream includes a combination of sparse space (reserved but not allocated) and allocated cluster data. The `$Max` stream contains all its data within the NTFS attribute and therefore is called *resident* data. Either of the data associated with these named streams can be copied by **ntfscopy**. Below are some examples:

```

ntfscopy c:$Extend$UsnJrnl:$J c:\dump\usnjrnl.bin
ntfscopy c:$Extend$UsnJrnl:$J c:\dump\usnjrnl.bin -skip_sparse_clusters
ntfscopy c:$Extend$UsnJrnl:$Max c:\dump\usnjrnl.max.bin
  
```

The first and second examples copy the *\$UsnJrnl:\$J* data stream, however, the second skips the sparse clusters and only copies the clusters that are allocated. **ntfscopy** handles sparse clusters as if they are clusters with zero content, therefore, the first example will substitute zeros in for each sparse cluster copied and the resulting file will be much larger, to the tune of approximately 0x153c0 hex clusters bigger, than the second without providing any additional value.

The third example shows one can target any stream within the file and copies out the *\$UsnJrnl:\$Max resident* data from the data attribute.

2.9 Copying Files from Volume Shadows

For starters, Volume Shadow copies, as is discussed here, only applies to Windows Vista, Win7, Win8 and beyond. It does not apply to Windows XP.

To tell **ntfscopy** to pull a file from a Volume Shadow, one specifies the option **-vss <index of volume shadow>**, and then uses the **-src** and **-dst** options to specify the source and the destination files, respectively. For example, to copy the *System* hive from the Volume Shadow with index 1 to the local drive, one uses the following syntax:

```
ntfscopy -vss 1 -src Windows\System32\Config\System -dst c:\dump\System
```

To determine which indexes are available from the various Volume Shadows, one can use the Windows built-in utility **vssadmin**, as follows:

```
vssadmin list shadows
```

To filter some of the extraneous detail, type

```
vssadmin list shadows | find /i "volume"
```

While the amount of data can be voluminous, the keywords one needs to look for are names that look like this:

```
Shadow Copy Volume: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1
```

```
Shadow Copy Volume: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy2
```

```
...
```

From the above, notice the number after the word *HarddiskVolumeShadowCopy*. It is this number that is passed as an argument with the option **-vss <index>**.

3 Known Issues

3.1 Linux and macOS Specific

When specifying directories that start with a dollar sign '\$', it needs to be escaped with a backslash. For example `$Extend` is a directory in the root directory of an NTFS partition. To list the contents of this directory, one needs to use the following notation:

```
<path>/ntfscopy / \ $MFT ./mft.bin -image < dd.image > -offset <vol offset>
```

The first slash in `/$MFT` specifies the root directory. The next character is a backslash to specify the '\$' is included after the first slash, and the rest of the characters are normal.

3.2 Windows Specific (Use of backslashes and quotes)

When passing in a directory and using quotes around the directory path, ensure you don't use a backslash at the end of the directory, since it will be interpreted as an escape sequence for the quote that follows:

```
ntfscopy "c:\windows" <== correct
```

```
ntfscopy "c:\windows\" <== incorrect, since the ending \" will be interpreted incorrectly.
```

4 Available Options

Option	Description
-partition	Extract artifacts from a mounted Windows volume. The syntax is -partition <drive letter> .
-vmdk	Extract artifacts from a VMWare monolithic NTFS formatted volume. The syntax is -vmdk <disk name> . For a collection of VMWare disks that include snapshots, one can use the following syntax: -vmdk "disk1 disk2 ..."
-drivenum	Extract artifacts from a mounted disk specified by a drive number and volume offset. The syntax is -drivenum <#> -offset <volume offset>
-image	Extract artifacts from a volume specified by an image and volume offset. The syntax is -image <filename> -offset <volume offset> .
-vss	Experimental. Extract artifacts from a Volume shadow. The syntax

	is -vss <index of volume shadow> . Only applies to Windows Vista, Win7, Win8 and beyond. Does not apply to Windows XP.
-raw	Output all the allocated clusters associated with the source file. This includes the slack space associated with the file.
-meta	Pull out the NTFS attributes associated with the source file. The output will be put into a separate file with the same name as the destination file with the suffix ".meta.txt" appended to the name.
-skip_sparse_clusters	Copy all allocated clusters that are not designated as sparse. This is useful when trying to copy the NTFS change log file, since many of the clusters in this file are sparse and are realized as zero.
-md5	Compute the MD5 hash of the file being copied and append the final hash to the name of the destination file. Also, the modification time in hexadecimal will be prepended to the front of the filename. This is useful when copying many files from multiple directories into one directory to ensure all files have a unique name.
-first_ntfsvol	Used in conjunction with the -image and -drivenum (Windows only) options. It is just a shorthand notation that says to find the offset of the first NTFS volume and use it as the source medium.
-src	To explicitly define the source file. The syntax is -src <filename> .
-dst	To explicitly define the destination file (or destination directory, if used in conjunction with the -pipe command). The syntax is -dst <filename or directory> .
-pipe	Used to pipe files into the tool via STDIN (standard input). Each file passed in is copied in sequence.
-enumdir	Experimental. Used to process files within a folder and/or subfolders. Each file is parsed in sequence. The syntax is -enumdir <folder> -num_subdirs <#> .
-filter	Filters data passed in via STDIN via the -pipe or -enumdir options. The syntax is -filter <"*.ext *partialname* ..."> . The wildcard character '*' is restricted to either before the name or after the name.
-cluster	Copy by logical cluster number (LCN). The syntax is -cluster <start

	LCN > <# <i>clusters</i> >. Logical, here, means it is referenced from a specific volume. It needs to be used in conjunction with a source medium identifier (eg. -partition (Windows only), -image , -vmdk , -drivenum (Windows only)).
-mft	Copy by MFT entry number. The syntax is -mft <#> . It needs to be used in conjunction with a source medium identifier (eg. -partition (Windows only), -image , -vmdk , -drivenum (Windows only)).

5 Authentication and the License File

This tool has authentication built into the binary. The primary authentication mechanism is the digital X509 code signing certificate embedded into the binary (Windows and macOS).

The other mechanism is the runtime authentication, which applies to all the versions of the tools (Windows, Linux and macOS). The runtime authentication ensures that the tool has a valid license. The license needs to be in the same directory of the tool for it to authenticate. Furthermore, any modification to the license, either to its name or contents, will invalidate the license.

5.1 *Limited* versus *Demo* versus *Full* in the tool's Output Banner

The tools from *TZWorks* will output header information about the tool's version and whether it is running in *limited*, *demo* or *full* mode. This is directly related to what version of a license the tool authenticates with. The *limited* and *demo* keywords indicates some functionality of the tool is not available, and the *full* keyword indicates all the functionality is available. The lacking functionality in the *limited* or *demo* versions may mean one or all of the following: (a) certain options may not be available, (b) certain data may not be outputted in the parsed results, and (c) the license has a finite lifetime before expiring.

6 References

1. <http://en.wikipedia.org/wiki/NTFS> website
2. Brian Carrier's book, File System Forensic Analysis, sections on NTFS
3. Various Microsoft TechNet articles.
4. VMWare Virtual Disk Format 1.1 Technical Note, <http://www.vmware.com>