SecureArtisan

My Road to Digital Forensics Excellence

# Windows 7 CD/DVD Burning

Posted by Paul Bobby on June 4, 2012

Plenty has been said about the “did they burn data to CD/DVD” question for Windows XP – but a recent case required me to answer the same question but this time for Windows 7. So has anything changed? Is it easier, harder? Well let’s find out.

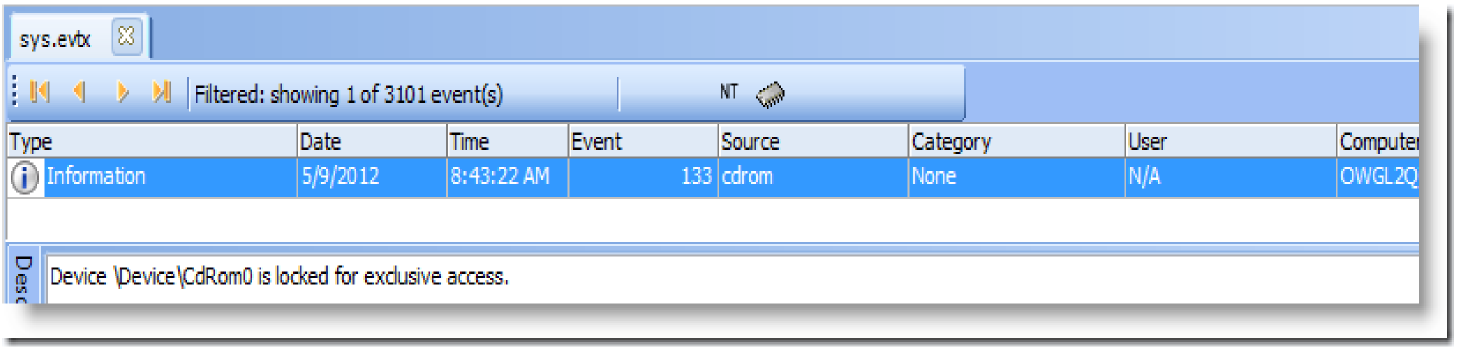
Windows 7 comes with built in support for burning CD’s and DVD’s. A tool called Windows DVD Maker is provided to end-users as part of the Windows 7 operating system. This guide highlights artifacts that indicate if/when a blank CD/DVD was inserted in to the drive, and also how to determine what was burned to that media.

## The Event Log

Windows 7 records information in 40+ event logs, not just the original big three (System, Application and Security). Several tests of burning data to DVD show that with under my current environment’s build and GPO configuration, the only event that gets written to a log is:

**Event ID 133. Source "cdrom"**

And this gets written to the *System Event Log only*. While there is no supporting data for this event at EventID.net or in the Microsoft Knowledge Base, testing has shown that this event occurs *when the burn button is clicked in Windows Explorer*. Simply adding non-blank discs does not trigger this event.



| Type | Date | Time | Event | Source | Category | User | Computer |
|---|---|---|---|---|---|---|---|
| Information | 5/9/2012 | 8:43:22 AM | 133 | cdrom | None | N/A | OWGL2Q |

Device \Device\CdRom0 is locked for exclusive access.

The description, "locked for exclusive access" shows that the '**burn**' process has actually started.

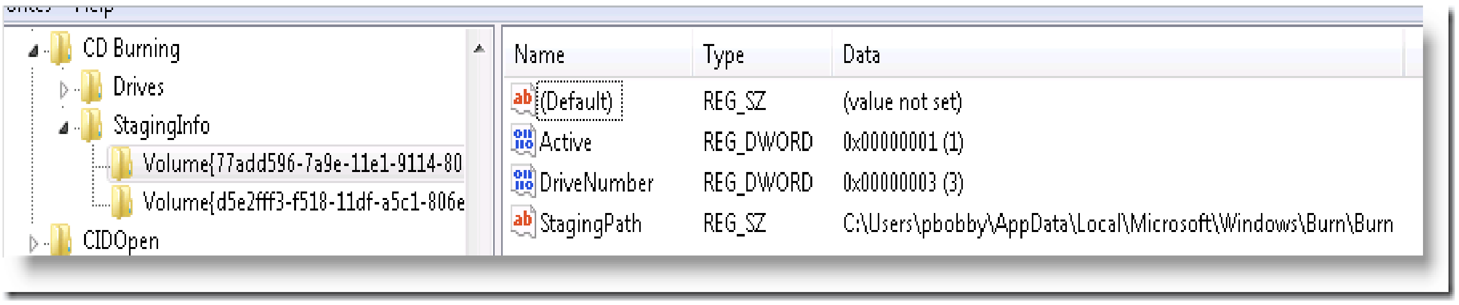
## The Registry

The following key:

> **HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\CD Burning\StagingInfo**

contains a KvP that resembles this:

> **Volume{77add596-7a9e-11e1-9114-806e6f6e6963}**


CD Burning
Drives
StagingInfo
Volume{77add596-7a9e-11e1-9114-80
Volume{d5e2fff3-f518-11df-a5c1-806e
CIDOpen

| Name | Type | Data |
|---|---|---|
| (Default) | REG_SZ | (value not set) |
| Active | REG_DWORD | 0x00000001 (1) |
| DriveNumber | REG_DWORD | 0x00000003 (3) |
| StagingPath | REG_SZ | C:\Users\pbobby\AppData\Local\Microsoft\Windows\Burn\Burn |

The value "StagingPath" is significant. It shows the folder on the volume that is used to 'stage' files prior to committing them to the disc.

Once we have a file on an NTFS volume to look at, we can resort to standard file system forensics to figure out what, if anything, existed in this folder. (Remember, even folders are files to NTFS)

# The Forensics

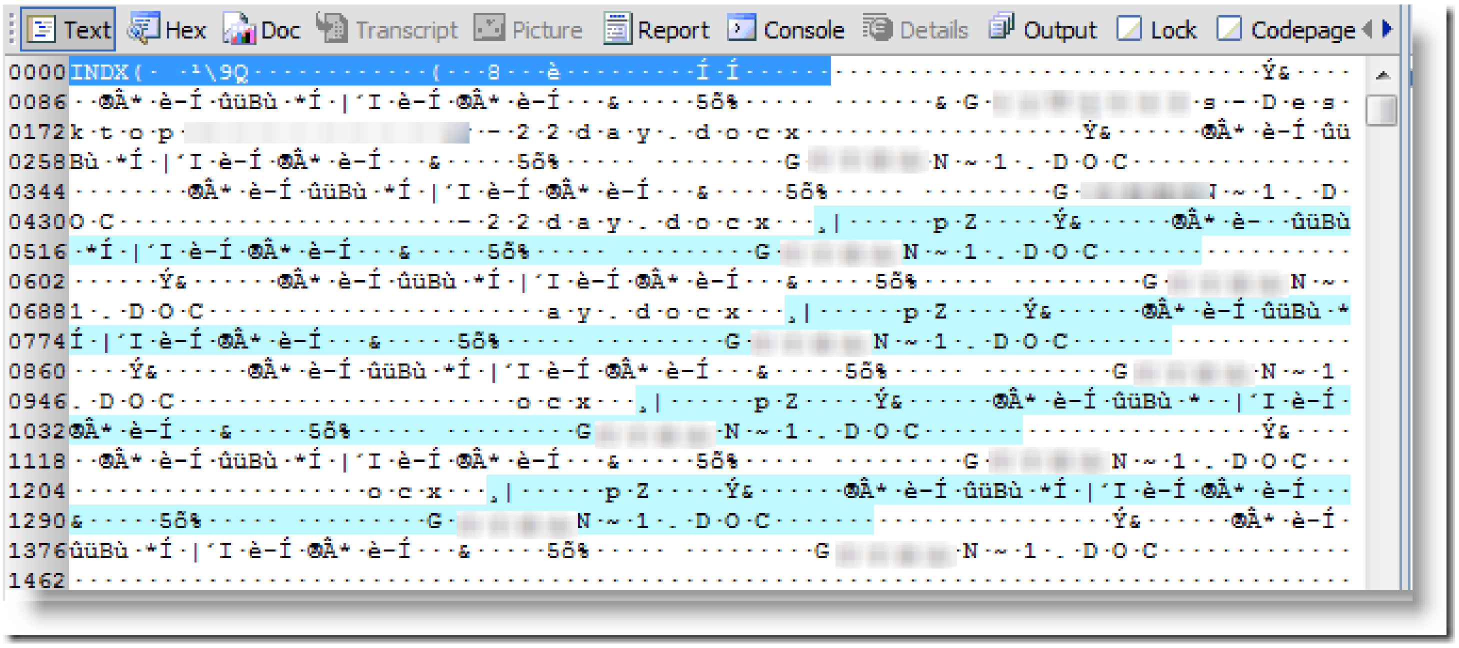
## Deleted Files

When parsing an NTFS volume using Encase, the tool automatically assigns all deleted files, with intact MFT records, to the appropriate parent folder.

In other words, when you navigate to this folder using Encase, you may see deleted files in that folder. If they are in the clear, then you got very lucky and found a burning session in progress. Any file present in this folder, clear or deleted, is/was staged there as part of a burning session.

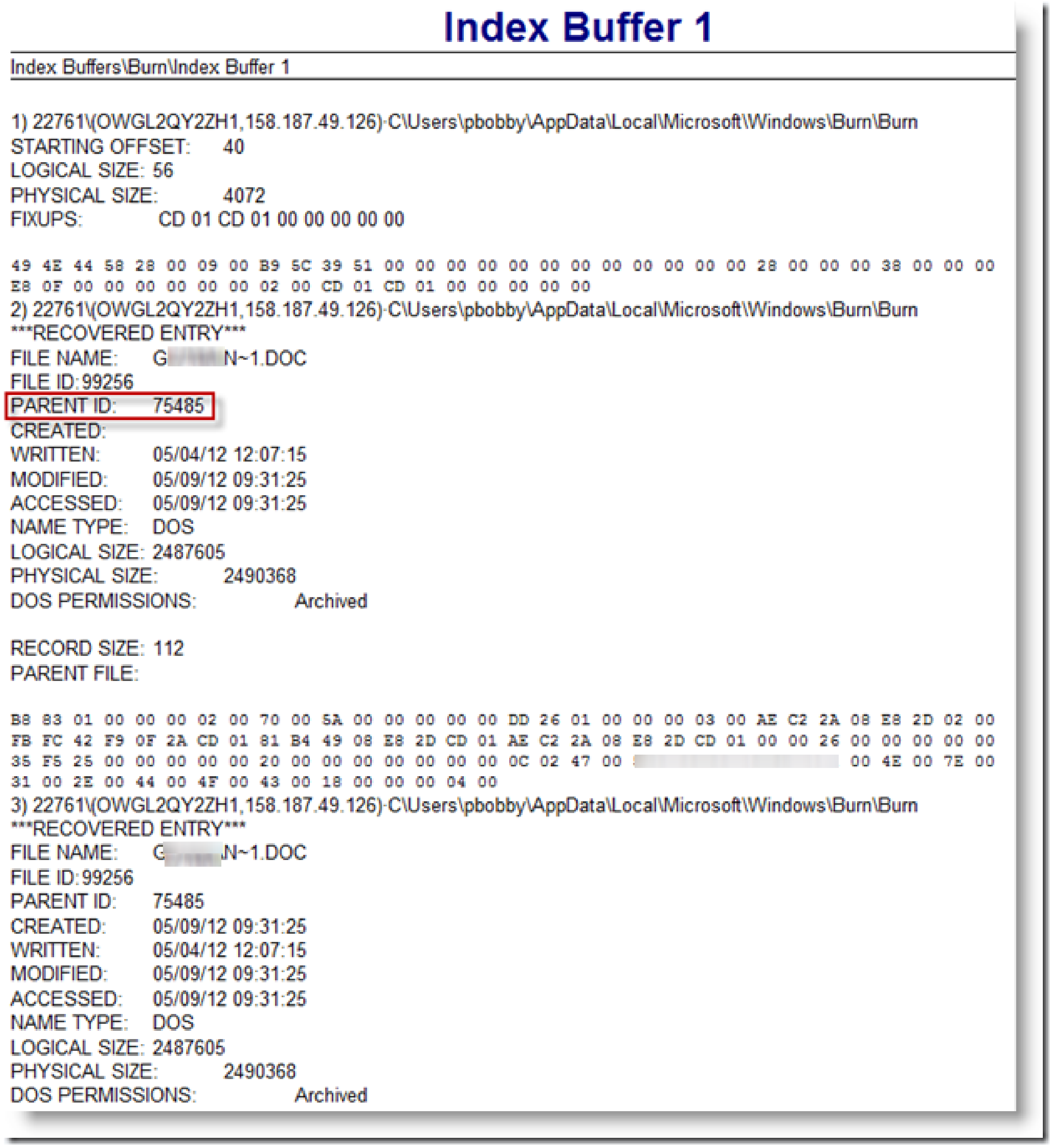
## INDX Data

Remember, every folder in NTFS is a file. What does the "file" for the "\burn" folder contain? It contains a binary record structure that the Windows OS uses to display the contents of that folder called an INDX record. Here's a sample:



```
0000INDX(·  ·¹\9Q·············(···8···è·········Í·Í······································Ý&·····
0086··®Â*·è–Í·ûüBù·*Í·|´I·è–Í·®Â*·è–Í···&·····5õ%·····  ·········&·G·[redacted]·s·–·D·e·s·
0172k·t·o·p·[redacted]·–·2·2·d·a·y·_·d·o·c·x·······················Ý&······®Â*·è–Í·ûü
0258Bù·*Í·|´I·è–Í·®Â*·è–Í···&·····5õ%·····  ·········G·[redacted]N·~·1·_·D·O·C················
0344·········®Â*·è–Í·ûüBù·*Í·|´I·è–Í·®Â*·è–Í···&·····5õ%·····  ···········G·[redacted]I·~·1·_·D·
0430O·C···························–·2·2·d·a·y·_·d·o·c·x···¸|······p·Z·····Ý&······®Â*·è–··ûüBù
0516·*Í·|´I·è–Í·®Â*·è–Í···&·····5õ%·····  ···········G·[redacted]N·~·1·_·D·O·C·····················
0602······Ý&······®Â*·è–Í·ûüBù·*Í·|´I·è–Í·®Â*·è–Í···&·····5õ%·····  ··········G·[redacted]N·~·
06881·_·D·O·C···························a·y·_·d·o·c·x···¸|······p·Z·····Ý&······®Â*·è–Í·ûüBù·*
0774Í·|´I·è–Í·®Â*·è–Í···&·····5õ%·····  ··········G·[redacted]N·~·1·_·D·O·C·······················
0860····Ý&······®Â*·è–Í·ûüBù·*Í·|´I·è–Í·®Â*·è–Í···&·····5õ%·····  ··········G·[redacted]N·~·1·
0946_·D·O·C·························o·c·x···¸|······p·Z·····Ý&······®Â*·è–Í·ûüBù·*··|´I·è–Í·
1032®Â*·è–Í···&·····5õ%·····  ··········G·[redacted]·N·~·1·_·D·O·C····················Ý&····
1118··®Â*·è–Í·ûüBù·*Í·|´I·è–Í·®Â*·è–Í···&·····5õ%·····  ··········G·[redacted]N·~·1·_·D·O·C···
1204························o·c·x···¸|······p·Z·····Ý&······®Â*·è–Í·ûüBù·*Í·|´I·è–Í·®Â*·è–Í···
1290&·····5õ%·····  ··········G·[redacted]N·~·1·_·D·O·C···························Ý&······®Â*·è–Í·
1376ûüBù·*Í·|´I·è–Í·®Â*·è–Í···&·····5õ%·····  ··········G·[redacted]N·~·1·_·D·O·C··············
1462·······························································································
```

The OS maintains the contents of this file on a real-time basis: as files are added to the folder, the content of this INDX is updated. As files are removed, the contents of this INDX are updated. Even though the folder is currently empty, it looks like there is some 'leftover' data in this file. Encase can parse this out for you using an enscript under Examples->Index Buffer Reader. Here's a partial screenshot of what the output can look like:



## Index Buffer 1

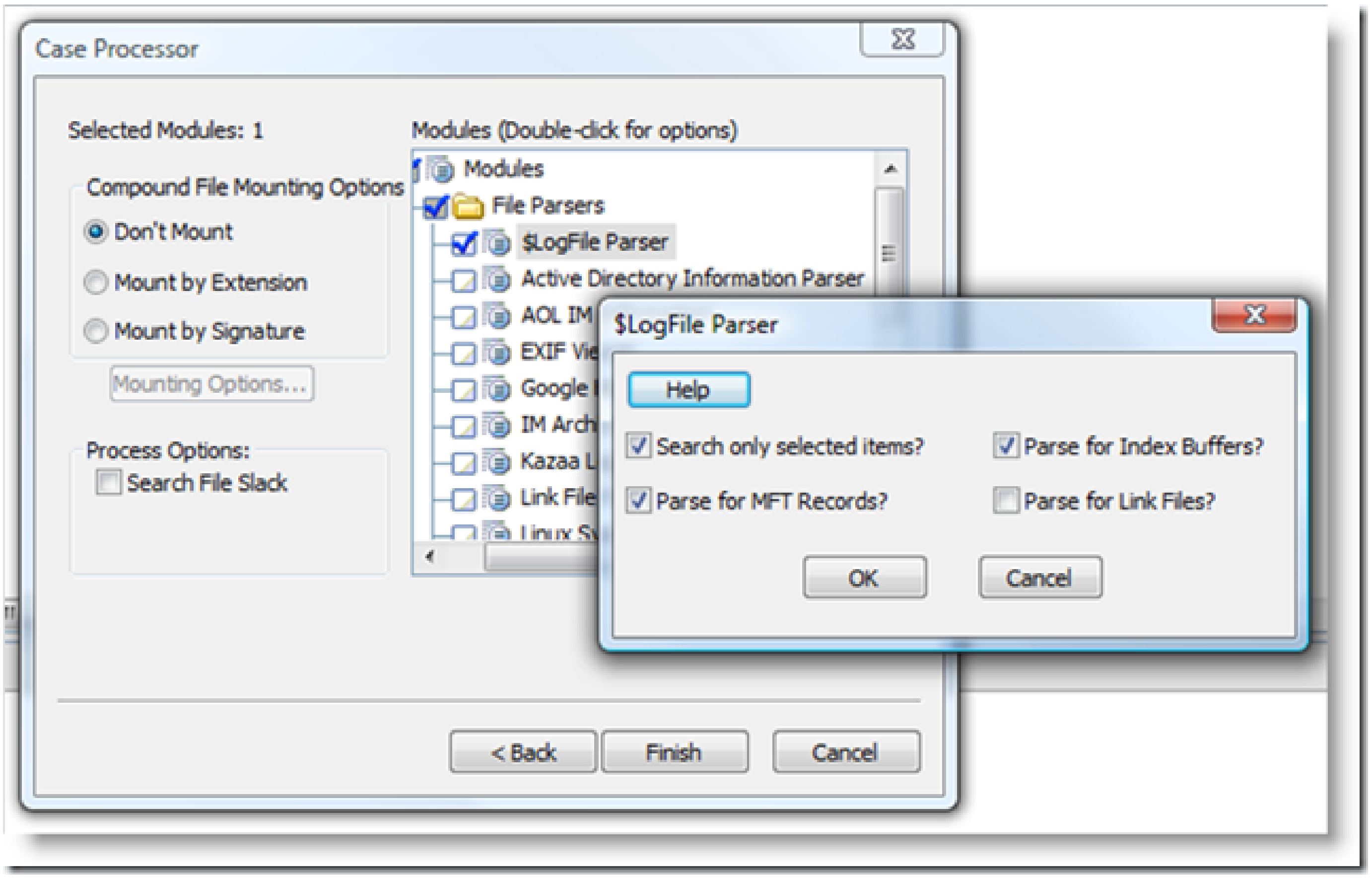
```
Index Buffers\Burn\Index Buffer 1

1) 22761\(OWGL2QY2ZH1,158.187.49.126)-C\Users\pbobby\AppData\Local\Microsoft\Windows\Burn\Burn
STARTING OFFSET:     40
LOGICAL SIZE: 56
PHYSICAL SIZE:       4072
FIXUPS:          CD 01 CD 01 00 00 00 00 00

49 4E 44 58 28 00 09 00 B9 5C 39 51 00 00 00 00 00 00 00 00 00 00 00 00 28 00 00 00 38 00 00 00
E8 0F 00 00 00 00 00 00 02 00 CD 01 CD 01 00 00 00 00 00
2) 22761\(OWGL2QY2ZH1,158.187.49.126)-C\Users\pbobby\AppData\Local\Microsoft\Windows\Burn\Burn
***RECOVERED ENTRY***
FILE NAME:     G[redacted]N~1.DOC
FILE ID:99256
PARENT ID:     75485
CREATED:
WRITTEN:       05/04/12 12:07:15
MODIFIED:      05/09/12 09:31:25
ACCESSED:      05/09/12 09:31:25
NAME TYPE:     DOS
LOGICAL SIZE: 2487605
PHYSICAL SIZE:       2490368
DOS PERMISSIONS:            Archived

RECORD SIZE: 112
PARENT FILE:

B8 83 01 00 00 00 02 00 70 00 5A 00 00 00 00 00 DD 26 01 00 00 00 03 00 AE C2 2A 08 E8 2D 02 00
FB FC 42 F9 0F 2A CD 01 81 B4 49 08 E8 2D CD 01 AE C2 2A 08 E8 2D CD 01 00 00 26 00 00 00 00 00
35 F5 25 00 00 00 00 00 20 00 00 00 00 00 00 00 0C 02 47 00 [redacted] 00 4E 00 7E 00
31 00 2E 00 44 00 4F 00 43 00 18 00 00 00 04 00
3) 22761\(OWGL2QY2ZH1,158.187.49.126)-C\Users\pbobby\AppData\Local\Microsoft\Windows\Burn\Burn
***RECOVERED ENTRY***
FILE NAME:     G[redacted]N~1.DOC
FILE ID:99256
PARENT ID:     75485
CREATED:       05/09/12 09:31:25
WRITTEN:       05/04/12 12:07:15
MODIFIED:      05/09/12 09:31:25
ACCESSED:      05/09/12 09:31:25
NAME TYPE:     DOS
LOGICAL SIZE: 2487605
PHYSICAL SIZE:       2490368
DOS PERMISSIONS:            Archived
```

The output shows that a file was present in this folder (you have a filename) and corresponding timestamps for that file show you when this occurred. It also comes with size information and MFT record information.

## MFT Records – $LogFile

What is the $Logfile? It contains information used by NTFS for faster recoverability. The log file is used by Windows Server 2003 to restore metadata consistency to NTFS after a system failure. The size of the log file depends on the size of the volume, but you can increase the size of the log file by using the Chkdsk command. Source Microsoft KB Article (The Logfile is of course used in more than just Server 2003).

This file contains several useful artifacts, but the one we want are the MFT records. Encase has an enscript to parse this file, it's under Case Processor.


Case Processor

Selected Modules: 1

Compound File Mounting Options
[✓ Don't Mount]
Mount by Extension
Mount by Signature

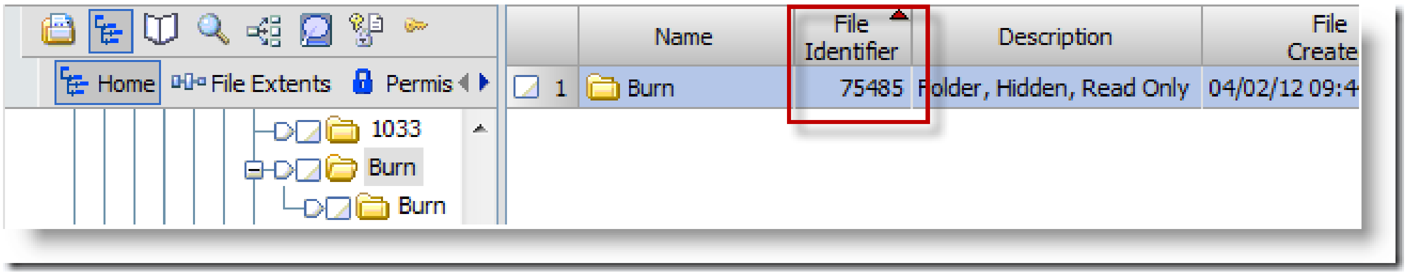
Process Options:
Search File Slack

Modules (Double-click for options)
- Modules
  - [✓] File Parsers
    - [✓] $LogFile Parser
    - [ ] Active Directory Information Parser
    - [ ] AOL IM
    - [ ] EXIF Vie
    - [ ] Google
    - [ ] IM Arch
    - [ ] Kazaa L
    - [ ] Link File
    - [ ] Linux S

$LogFile Parser
- [✓] Search only selected items?
- [✓] Parse for Index Buffers?
- [✓] Parse for MFT Records?
- [ ] Parse for Link Files?

Once parsed, your case will contain a series of bookmarks (sometimes 100s of the things), one bookmark for each MFT record found in the $LogFile. What do we want to look for in these bookmarks?

MFT records contains lots of useful information, but the piece we are looking for is called the "Parent MFT Record #". This associates a file with a particular parent folder, and that is basically how things are kept organized. So let's find the MFT record for the "\Burn" folder.



| | Name | File Identifier | Description | File Create |
|---|---|---|---|---|
| 1 | Burn | 75485 | Folder, Hidden, Read Only | 04/02/12 09:4 |

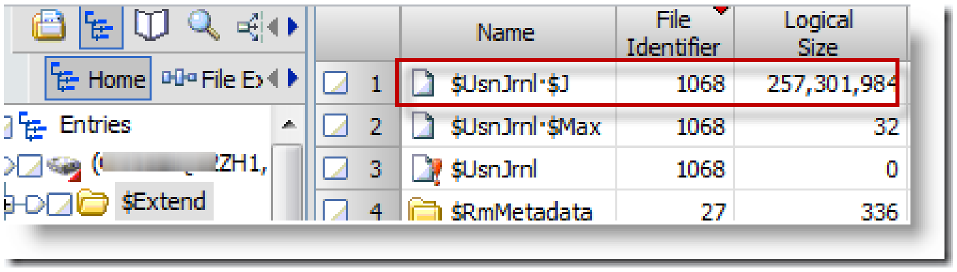
The MFT record # for the Burn folder is 75485. Note make sure you select the right folder. You want the MFT record for that second 'burn' folder, not the first.

So, make your own Condition under the Bookmarks view called "Comment contains" that finds any comment that contains the parent MFT record #, and ta-da, you now have a list of files that once existed in that folder. Why? Because every MFT record also stores the MFT number of its parent folder.

## MFT Records – The Journal File

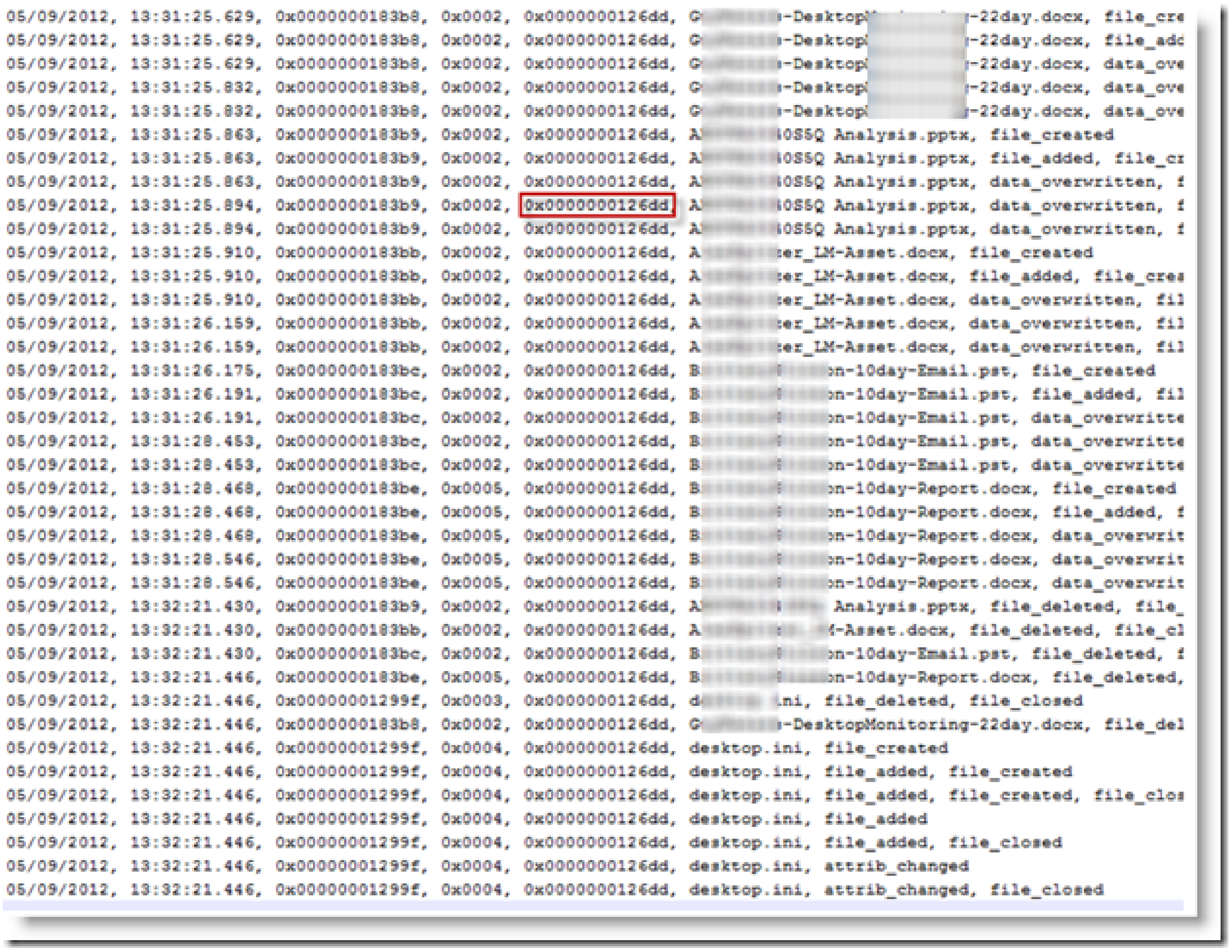
What is this journal file? The USN Journal (Update Sequence Number Journal) is a system management feature that records changes to all files, streams and directories on the volume, as well as their various attributes and security settings. The journal is made available for applications to track changes to the volume.[12] This journal can be enabled or disabled on non-system volumes[13] and is not enabled by default for a newly added drive. Source Wikipedia article on NTFS

This information is stored in a file called $USNJRNL, in the folder C:\$Extend. The files comes with two streams, and the one that contains the data we really want is:



| | Name | File Identifier | Logical Size |
|---|---|---|---|
| 1 | $UsnJrnl·$J | 1068 | 257,301,984 |
| 2 | $UsnJrnl·$Max | 1068 | 32 |
| 3 | $UsnJrnl | 1068 | 0 |
| 4 | $RmMetadata | 27 | 336 |

Encase does not have a built-in enscript for parsing that file, but there are some scripts out there, along with standalone tools that can parse the Journal file. My favorite comes from TZWorks and is called Windows Journal Parser. Get it. Here is a sample output from this tool, already filtered to include files whose parent MFT ID matches that of the "\burn" folder:



```
05/09/2012, 13:31:25.629, 0x00000000183b8, 0x0002, 0x0000000126dd, G[redacted]-Desktop[redacted]-22day.docx, file_cre
05/09/2012, 13:31:25.629, 0x00000000183b8, 0x0002, 0x0000000126dd, G[redacted]-Desktop[redacted]-22day.docx, file_adc
05/09/2012, 13:31:25.629, 0x00000000183b8, 0x0002, 0x0000000126dd, G[redacted]-Desktop[redacted]-22day.docx, data_ove
05/09/2012, 13:31:25.832, 0x00000000183b8, 0x0002, 0x0000000126dd, G[redacted]-Desktop[redacted]-22day.docx, data_ove
05/09/2012, 13:31:25.832, 0x00000000183b8, 0x0002, 0x0000000126dd, G[redacted]-Desktop[redacted]-22day.docx, data_ove
05/09/2012, 13:31:25.863, 0x00000000183b9, 0x0002, 0x0000000126dd, A[redacted]0S5Q Analysis.pptx, file_created
05/09/2012, 13:31:25.863, 0x00000000183b9, 0x0002, 0x0000000126dd, A[redacted]0S5Q Analysis.pptx, file_added, file_cr
05/09/2012, 13:31:25.863, 0x00000000183b9, 0x0002, 0x0000000126dd, A[redacted]0S5Q Analysis.pptx, data_overwritten, f
05/09/2012, 13:31:25.894, 0x00000000183b9, 0x0002, 0x0000000126dd, A[redacted]0S5Q Analysis.pptx, data_overwritten, f
05/09/2012, 13:31:25.894, 0x00000000183b9, 0x0002, 0x0000000126dd, A[redacted]0S5Q Analysis.pptx, data_overwritten, f
05/09/2012, 13:31:25.910, 0x00000000183bb, 0x0002, 0x0000000126dd, A[redacted]er_LM-Asset.docx, file_created
05/09/2012, 13:31:25.910, 0x00000000183bb, 0x0002, 0x0000000126dd, A[redacted]er_LM-Asset.docx, file_added, file_crea
05/09/2012, 13:31:25.910, 0x00000000183bb, 0x0002, 0x0000000126dd, A[redacted]er_LM-Asset.docx, data_overwritten, fil
05/09/2012, 13:31:26.159, 0x00000000183bb, 0x0002, 0x0000000126dd, A[redacted]er_LM-Asset.docx, data_overwritten, fil
05/09/2012, 13:31:26.159, 0x00000000183bb, 0x0002, 0x0000000126dd, A[redacted]er_LM-Asset.docx, data_overwritten, fil
05/09/2012, 13:31:26.175, 0x00000000183bc, 0x0002, 0x0000000126dd, B[redacted]on-10day-Email.pst, file_created
05/09/2012, 13:31:26.191, 0x00000000183bc, 0x0002, 0x0000000126dd, B[redacted]on-10day-Email.pst, file_added, fil
05/09/2012, 13:31:26.191, 0x00000000183bc, 0x0002, 0x0000000126dd, B[redacted]on-10day-Email.pst, data_overwritte
05/09/2012, 13:31:28.453, 0x00000000183bc, 0x0002, 0x0000000126dd, B[redacted]on-10day-Email.pst, data_overwritte
05/09/2012, 13:31:28.453, 0x00000000183bc, 0x0002, 0x0000000126dd, B[redacted]on-10day-Email.pst, data_overwritte
05/09/2012, 13:31:28.468, 0x00000000183be, 0x0005, 0x0000000126dd, B[redacted]on-10day-Report.docx, file_created
05/09/2012, 13:31:28.468, 0x00000000183be, 0x0005, 0x0000000126dd, B[redacted]on-10day-Report.docx, file_added, f
05/09/2012, 13:31:28.468, 0x00000000183be, 0x0005, 0x0000000126dd, B[redacted]on-10day-Report.docx, data_overwrit
05/09/2012, 13:31:28.546, 0x00000000183be, 0x0005, 0x0000000126dd, B[redacted]on-10day-Report.docx, data_overwrit
05/09/2012, 13:31:28.546, 0x00000000183be, 0x0005, 0x0000000126dd, B[redacted]on-10day-Report.docx, data_overwrit
05/09/2012, 13:32:21.430, 0x00000000183b9, 0x0002, 0x0000000126dd, A[redacted] Analysis.pptx, file_deleted, file_
05/09/2012, 13:32:21.430, 0x00000000183bb, 0x0002, 0x0000000126dd, A[redacted]M-Asset.docx, file_deleted, file_cl
05/09/2012, 13:32:21.430, 0x00000000183bc, 0x0002, 0x0000000126dd, B[redacted]on-10day-Email.pst, file_deleted, f
05/09/2012, 13:32:21.446, 0x00000000183be, 0x0005, 0x0000000126dd, B[redacted]on-10day-Report.docx, file_deleted,
05/09/2012, 13:32:21.446, 0x000000001299f, 0x0003, 0x0000000126dd, d[redacted]ini, file_deleted, file_closed
05/09/2012, 13:32:21.446, 0x00000000183b8, 0x0002, 0x0000000126dd, G[redacted]-DesktopMonitoring-22day.docx, file_del
05/09/2012, 13:32:21.446, 0x000000001299f, 0x0004, 0x0000000126dd, desktop.ini, file_created
05/09/2012, 13:32:21.446, 0x000000001299f, 0x0004, 0x0000000126dd, desktop.ini, file_added, file_created
05/09/2012, 13:32:21.446, 0x000000001299f, 0x0004, 0x0000000126dd, desktop.ini, file_added, file_created, file_clos
05/09/2012, 13:32:21.446, 0x000000001299f, 0x0004, 0x0000000126dd, desktop.ini, file_added
05/09/2012, 13:32:21.446, 0x000000001299f, 0x0004, 0x0000000126dd, desktop.ini, file_added, file_closed
05/09/2012, 13:32:21.446, 0x000000001299f, 0x0004, 0x0000000126dd, desktop.ini, attrib_changed
05/09/2012, 13:32:21.446, 0x000000001299f, 0x0004, 0x0000000126dd, desktop.ini, attrib_changed, file_closed
```

The highlighted piece is the hex value for the parent MFT. Any entry that has this same parent MFT hex value once existed in this folder, therefore the data was burned, or staged for burning.

## So that's it? Any problems?

One thing I can think of, what if the user starts staging data in that folder but changes his mind and deletes it? Can I prove the actual burn took place? It looks like Event ID 133 occurs when the burn button is clicked, so if you can correlate this event to MFT records showing content in that folder, then it's a solid conclusion to make. And I'm beginning to suspect that staging is not the write word to use. Continue reading.

If you look at the output of the USNJRNL parse, you'll see that the file was copied and deleted all within the same second – I'd like to think that that indicates a successful burn. Some further testing is needed there – but it appears that staging doesn't occur as I traditionally think of staging: as you select files and folders, place that data in the burn folder until you click Burn. That's inefficient, instead it looks some sort of data structure is maintained in RAM to hold pointers to the data you wish to burn, and the actual commit to disk process involves taking each file, one at a time, and writing to the Burn folder, then to Disc, then deleting from the Burn folder.

I welcome any additional information you may have, especially if you have noticed different behavior or discrepancies in my write up.

- 
- 
- 
- 
- http://bit.ly/1bC6727

This entry was posted on June 4, 2012 at 4:31 pm and is filed under Forensics. You can follow any responses to this entry through the RSS 2.0 feed. You can leave a response, or trackback from your own site.

« CEIC2012 Part 2
The malware made you do it? »