NTFS Log Tracker

blueangel

blueangel1275@gmail.com

forensic-note.blogspot.kr

Junghoon Oh

Index

Introduction

Introduction

- **NTFS's Log File**
  - $LogFile : Transaction Log
  - $UsnJrnl : Change Log

- Conventional file system forensics for NTFS
  - File system event based analysis primarily focusing on $MFT
    - ✓ $MFT : A file containing meta data for all files and directories in NTFS.
  - For deleted files it is possible that there is no meta data in $MFT
    - ✓ Finding artifacts of deleted is very difficulty for the following reasons
      - In case of system drive(C:), the OS creates temp files constantly.
      - A periodic garbage collection since Windows 7.
      - In case of SSD, unallocated space is arranged by TRIM operation.

Introduction

- **Analysis of $LogFile and $UsnJrnl**
  - With these files, an investigator can analyze the file system events during a specific period.
  - The file system events that are not in $MFT can still be analyzed
    - ✓ The history of deleted file
    - ✓ The history of a specific file – $MFT provides only last modified/access time of a file.
      - Identify history of access time of a particular file.
      - Identify history of modified time of a particular file.

$LogFile

- **$LogFile ?**
- The Structure of $LogFile
- The Event Analysis of $LogFile

$LogFile ?

- **The transaction log file of NTFS**
  - In case of unexpected system shutdown due to power error or critical system failure, the operating system recovers the status of file system to the previous status with saved information in "$LogFile" file.
  - $LogFile contains all file system transaction records.
    - ✓ The creation of file/directory
    - ✓ The deletion of file/directory
    - ✓ The modification of $data
    - ✓ The modification of MFT entry
  - Each record has LSN($LogFile Sequence Number).
    - ✓ This LSN information increase sequentially.
  - Each record has the operation data and the data before operation for restoration
    - ✓ Redo : The data after operation
    - ✓ Undo : The data before operation
  - Each volume has $LogFile.
  - It is located at entry number 2 of MFT.

Entry Number	File Name	Stored Information
0	$MFT	MFT Entry
1	$MFTMirr	Backup of $MFT
2	**$LogFile**	**Transaction Log**
3	$Volume	Volume label, Identifier, Version

$LogFile ?

- **Size of $LogFile**
  - 64 MB in typical hard disk volume.
  - The size can changed based on volume size but typically it is less than 64 MB.
  - In case of typical computer usage (web surfing, working on documents, etc), the capacity of 64 MB can hold 2 ~ 3 hours of activities in $LogFile records.
  - For forensic readiness, the size of the file should be increased.
- **Resize of $LogFile**
  - chkdsk /L ➔ Print current file size
  - "/L : [filesize(KB)]" ➔ Modification of file size

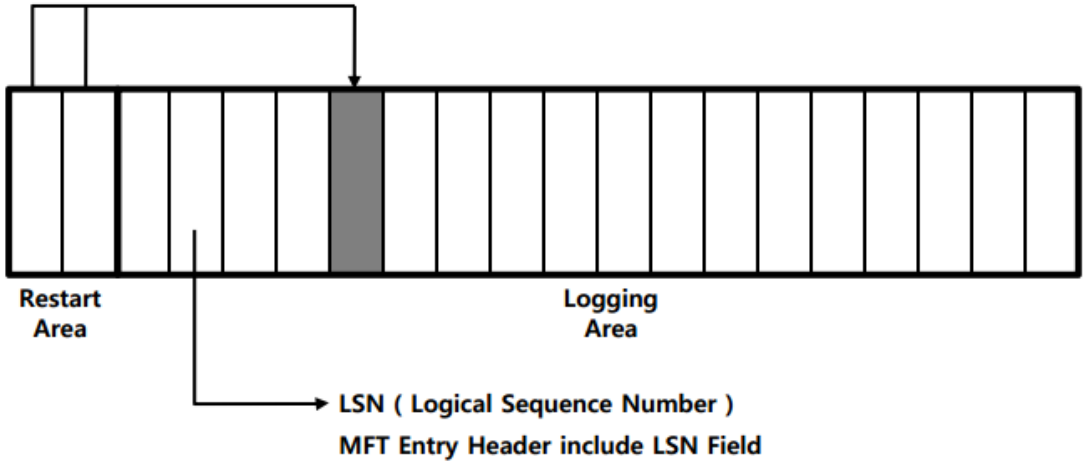
```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>chkdsk /L
The type of the file system is NTFS.
The current log file size is 65536 KB.
The default log file size for this volume is 65536 KB.
```

$LogFile

- $LogFile ?
- **The Structure of $LogFile**
- The Event Analysis of $LogFile

Overall Structure

- **Restart Area and Logging Area**
  - The basic unit of each area is a page.(size : 0x1000)
  - Restart Area
    - ✓ This area has information of the last operation, also known as current operation, record.
    - ✓ The location of restart area is first and second page (0x0000~0x2000) in the $LogFile.
  - Logging Area
    - ✓ This area has actual operation records.
    - ✓ It is located after "Restart Area"(0x2000~)
    - ✓ It is divided into "Buffer Page Area" and "Normal Page Area"



Restart Area

- **The information of the last or current operation record**
  - The "Current LSN" has the LSN information of the last operation record.

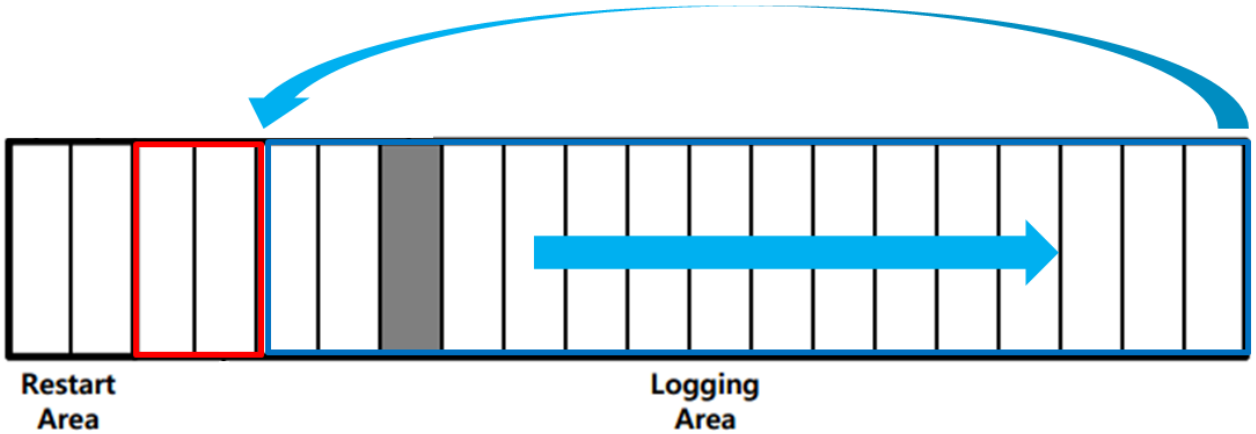
- **Two consecutive pages, second page is for the backup**
  - Each page starts with the magic number(RSTR).

- **The format of Restart Area**

0 1 2 3	4 5	6 7	8 9	A B	C D	E F
"RSTR" (Magic Number)	Update Sequence Offset	Update Sequence Count	Check Disk LSN			
System Page Size	Log Page Size		Restart Offset	Minor Version	Major Version	
Update Sequence Array						
Current LSN			Log Client	Client List	Flags	

Logging Area

- **The actual operation records are recorded.**
- **This area is divided into "Buffer Page Area" and "Normal Page Area"**
  - Buffer Page Area
    - ✓ The first two pages (0x2000~0x4000) in Logging Area. The second page is for the backup purpose.
    - ✓ The operation records are stored sequentially.
    - ✓ If the page is full of records, the content of page is moved to "Normal Page Area"
    - ✓ The last operation, therefore, record is stored in this area.
  - Normal Page Area
    - ✓ The rest of the logging area except for "Buffer Page Area"(0x4000~)
    - ✓ The operation records are stored sequentially.
    - ✓ If the area is full of records, the records are overwritten from the start of area.



The Structure of Page

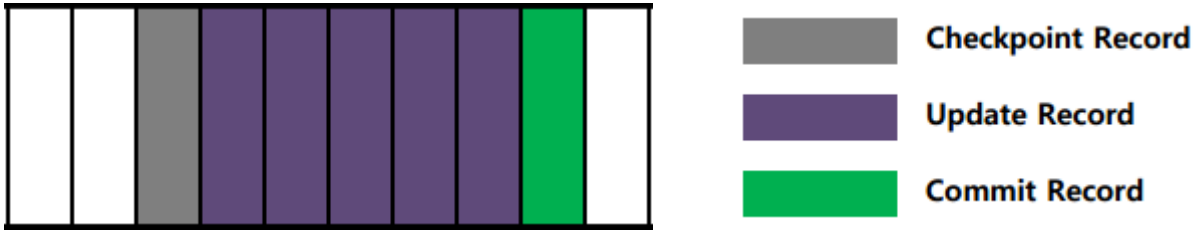
- **Page Configuration**
  - One header and multiple operation records
  - If the last operation record does not fit in a page, the rest of the record contents are stored in the next page continuously.

- **Page Header : the meta data of page is stored.**
  - Magic Number : "RCRD"
  - Last LSN : the highest LSN among the records including the record of crossed the page.
  - Next Record Offset : the offset of record having the highest LSN in page.
  - Last End LSN : the highest LSN among the records except record that crossed the page.

<table>
<tr><th>0 1 2 3</th><th>4 5</th><th>6 7</th><th>8 9</th><th>A B</th><th>C D E F</th></tr>
<tr><td>"RCRD"
(Magic Number)</td><td>Update
Sequence
Offset</td><td>Update
Sequence
Count</td><td colspan="3">Last LSN or File Offset</td></tr>
<tr><td>Flags</td><td>Page
Count</td><td>Page
Position</td><td>Next
Record
Offset</td><td>Word
Align</td><td>DWord Align</td></tr>
<tr><td colspan="3">Last End LSN</td><td colspan="3"></td></tr>
<tr><td colspan="6">Update Sequence Array</td></tr>
</table>

The Structure of Operation Record

- **Operation Record**
  - The actual content of transaction operation is stored.
  - A transaction operation is consist of multiple operation records sequentially.
    - ✓ Check Point Record : the start record of transaction
    - ✓ Update Record : the middle records of transaction
    - ✓ Commit Record : the last record of transaction

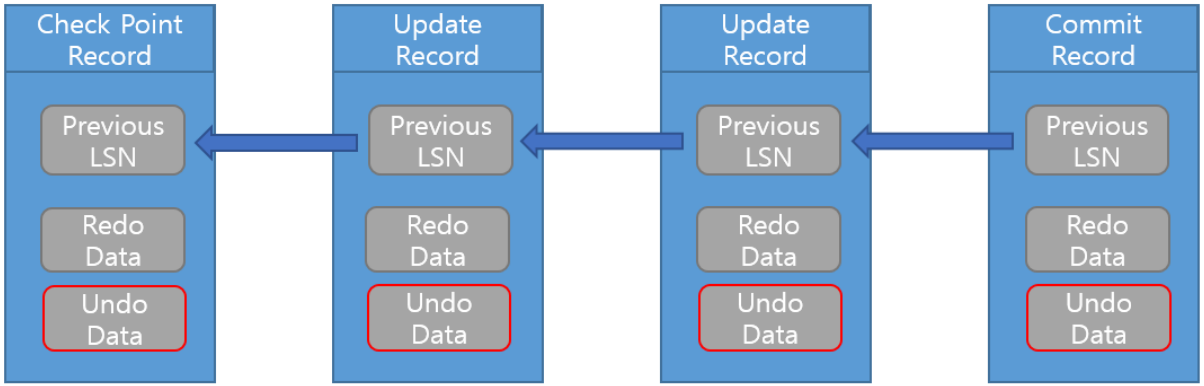


  - All operation records have the information of previous operation record except "Check Point Record".

The Structure of Operation Record

- **Operation Record(continue…)**
  - Configuration of Operation Record : Header + Data
    - ✓ Header : the meta data of record, Fixed Size(0x58)
    - ✓ Data
      - Redo : The data after operation finished (Example: the written data for 'write' operation)
      - Undo : The data before operation(Example: the data before 'write' operation started)

  - **The workflow when error recovery is executed**
    - ✓ OS performs backtracking from "Commit Record" with "Previous LSN" and applies "Undo" data.



The Structure of Operation Record

▪ The Format of Operation Record

- This LSN : LSN of current record
- Previous LSN : LSN of previous record
- Client Undo LSN : In case of a error recovery, a LSN information of record has following 'Undo' operation.
- Client Data Length : Size of Record(from "Redo Op" field to end of the record)
- Record Type : 0x02 (Check Point Record), 0x01(the rest Record)
- Flags : 0x01(record cross the current page), 0x00(record doesn't cross the current page)

0	1	2	3	4	5	6	7	8	9	A	B	C
This LSN						Previous LSN						
Client Undo LSN						Client Data Length				Client ID		
Record Type			Transaction ID			Flags	Alignment or Reserved					
Redo OP	Undo OP		Redo Offset		Redo Length	Undo Offset		Undo Length		Target Attribute		LCNs to follows
Record Offset	Attr Offset		MFT Cluster Index		Alignment or Reserved	Target VCN				Alignment or Reserved		
Target LCN			Alignment or Reserved									

The Structure of Operation Record

- **The Format of Operation Record(continue…)**
  - Redo Op : Redo operation code
  - Undo Op : Undo operation code
  - Redo Offset : Offset of "Redo" data(from "Redo Op" field)
  - Redo Length : Size of "Redo" data
  - Undo Offset : Offset of "Undo" data(from "Redo Op" field)
  - Undo Length : Size of "Undo" data

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
This LSN							Previous LSN								
Client Undo LSN							Client Data Length			Client ID					
Record Type			Transaction ID				Flags	Alignment or Reserved							
Redo OP	Undo OP		Redo Offset		Redo Length		Undo Offset		Undo Length		Target Attribute		LCNs to follows		
Record Offset	Attr Offset		MFT Cluster Index		Alignment or Reserved		Target VCN				Alignment or Reserved				
Target LCN			Alignment or Reserved												

The Structure of Operation Record

- **The Format of Operation Record(continue…)**
  - LCNs to Follows : 0x01(There is a next record), 0x00(There is no next record)
  - Record Offset
    - ✓ In case of operation to MFT record, the offset of attribute applied Redo/Undo data within the MFT record.
    - ✓ In case of the rest operation, the value is 0x00
  - Attr Offset
    - ✓ In case of operation to MFT record, the offset of point applied Redo/ Undo data within the attribute
    - ✓ In case of other operation, the offset of point applied Redo/Undo data within the cluster

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
This LSN						Previous LSN									
Client Undo LSN						Client Data Length				Client ID					
Record Type			Transaction ID			Flags		Alignment or Reserved							
Redo OP	Undo OP		Redo Offset		Redo Length	Undo Offset		Undo Length		Target Attribute		LCNs to follows			
Record Offset	Attr Offset		MFT Cluster Index		Alignment or Reserved	Target VCN				Alignment or Reserved					
Target LCN			Alignment or Reserved												

The Structure of Operation Record

▪ The Format of Operation Record(continue…)

- MFT Cluster Index : In case of operation for MFT record, the location of record applied Redo/Undo data within cluster
  - ✓ First (0x0000), Second(0x0002), Third (0x0003), forth(0x0006)
- Target VCN : VCN(Virtual Cluster Number) of "$MFT" file applied Redo/Undo data
- Target LCN : LCN(Logical Cluster Number) of the disk applied Redo/Undo data

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
This LSN						Previous LSN									
Client Undo LSN						Client Data Length				Client ID					
Record Type			Transaction ID			Flags		Alignment or Reserved							
Redo OP	Undo OP		Redo Offset		Redo Length	Undo Offset		Undo Length		Target Attribute		LCNs to follows			
Record Offset	Attr Offset		MFT Cluster Index		Alignment or Reserved	Target VCN				Alignment or Reserved					
Target LCN			Alignment or Reserved												

The Structure of Operation Record

- **Redo/Undo Operation Code**

NTFS Operation	Hex Value
Noop	0x00
CompensationlogRecord	0x01
InitializeFileRecordSegment	0x02
DeallocateFileRecordSegment	0x03
WriteEndofFileRecordSegement	0x04
CreateAttribute	0x05
DeleteAttribute	0x06
UpdateResidentValue	0x07
UpdataeNonResidentValue	0x08
UpdateMappingPairs	0x09
DeleteDirtyClusters	0x0A
SetNewAttributeSizes	0x0B

The Structure of Operation Record

- **Redo/Undo Operation Code(continue…)**

AddindexEntryRoot	0x0C
DeleteindexEntryRoot	0x0D
AddIndexEntryAllocation	0x0F
SetIndexEntryVenAllocation	0x12
UpdateFileNameRoot	0x13
UpdateFileNameAllocation	0x14
SetBitsInNonresidentBitMap	0x15
ClearBitsInNonresidentBitMap	0x16
PrepareTransaction	0x19
CommitTransaction	0x1A
ForgetTransaction	0x1B
OpenNonresidentAttribute	0x1C
DirtyPageTableDump	0x1F
TransactionTableDump	0x20
UpdateRecordDataRoot	0x21

$LogFile

- $LogFile ?
- The Structure of $LogFile
- **The Event Analysis of $LogFile**

The Event Analysis of $LogFile

- **The need for event analysis based on file-level events**
  - The information stored in an operation record is not based on file-level events
    - ✓ A transaction operation is consist of multiple operation records sequentially.
  - Creating a need for transforming information to file-level events which is meaningful to investigator.
  - The file-level events focused on this research are
    - ✓ Creating File/Directory
    - ✓ Deleting File/Directory
    - ✓ Writing Data
    - ✓ Renaming File/Directory
    - ✓ Moving File/Directory

Creating File/Directory

- **Creating Resident File**

LSN	Previous LSN	Record Type	Event	Detail	Fila Name	Full Path	Redo	Undo
8404761	0	Update Record					OpenNonresidentAttribute	Noop
8404778	8404761	Update Record					Set Bits In Nonresident Bit Map	Clear Bits In Nonresident Bit Map
8404790	8404778	Update Record					Noop	Deallocate File Record Segment
8404801	8404790	Update Record					OpenNonresidentAttribute	Noop
8404819	8404801	Update Record					Add Index Entry Allocation	Delete Index Entry Allocation
8404843	8404819	Update Record					Initialize File Record Segment	Noop
8404891	8404843	Commit Record					Forget Transaction	Compensation Log Record

LSN	Previous LSN	Record Type	Event	Detail	Fila Name	Full Path	Redo	Undo
8405882	0	Check Point Record					Noop	Noop
8405901	0	Update Record					Set Bits In Nonresident Bit Map	Clear Bits In Nonresident Bit Map
8405913	8405901	Update Record					Noop	Deallocate File Record Segment
8405924	8405913	Update Record					Add Index Entry Allocation	Delete Index Entry Allocation
8405948	8405924	Update Record					Initialize File Record Segment	Noop
8405996	8405948	Commit Record					Forget Transaction	Compensation Log Record

LSN	Previous LSN	Record Type	Event	Detail	Fila Name	Full Path	Redo	Undo
8406985	0	Check Point Record					Noop	Noop
8407004	0	Update Record					Set Bits In Nonresident Bit Map	Clear Bits In Nonresident Bit Map
8407016	8407004	Update Record					Noop	Deallocate File Record Segment
8407027	8407016	Update Record					Add Index Entry Allocation	Delete Index Entry Allocation
8407059	8407027	Update Record					Initialize File Record Segment	Noop
8407107	8407059	Commit Record					Forget Transaction	Compensation Log Record

- The record order of creating resident file(Redo/Undo)
  1. 0x15/0x16(Set Bits In Nonresident Bit Map/Clear Bits In Nonresident Bit Map)
  2. 0x00/0x03(Noop/Deallocate File Record Segment)
  3. 0x0E/0x0F(Add Index Entry Allocation/Delete Index Entry Allocation)
  4. 0x02/0x00(Initialize File Record Segment/Noop)
  5. 0x1B/0x01(Forget Transaction/Compensation Log Record)

- **The above screen shot is taken from a research version of $LogFile parsing tool.**

Creating File/Directory

- **The Information that can be obtained from a resident file creation event**
  - MFT Entry Number
    - ✓ From Redo data of 0x15/0x16(Set Bits In Nonresident Bit Map/Clear Bits In Nonresident Bit Map) operation record
    - ✓ The first four bytes of Redo data is "MFT Entry Number" of targeted MFT record by operation

```
0001F950  2A 3F 80 00 00 00 00 00  19 3F 80 00 00 00 00 00
0001F960  19 3F 80 00 00 00 00 00  30 00 00 00 00 00 00 00
0001F970  01 00 00 00 18 00 00 00  00 00 00 00 00 00 00 00
0001F980  15 00 16 00 28 00 08 00  28 00 08 00 C8 00 01 00
0001F990  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0001F9A0  FF FF 03 00 00 00 00 00  23 00 00 00 01 00 00 00
```

Current LSN

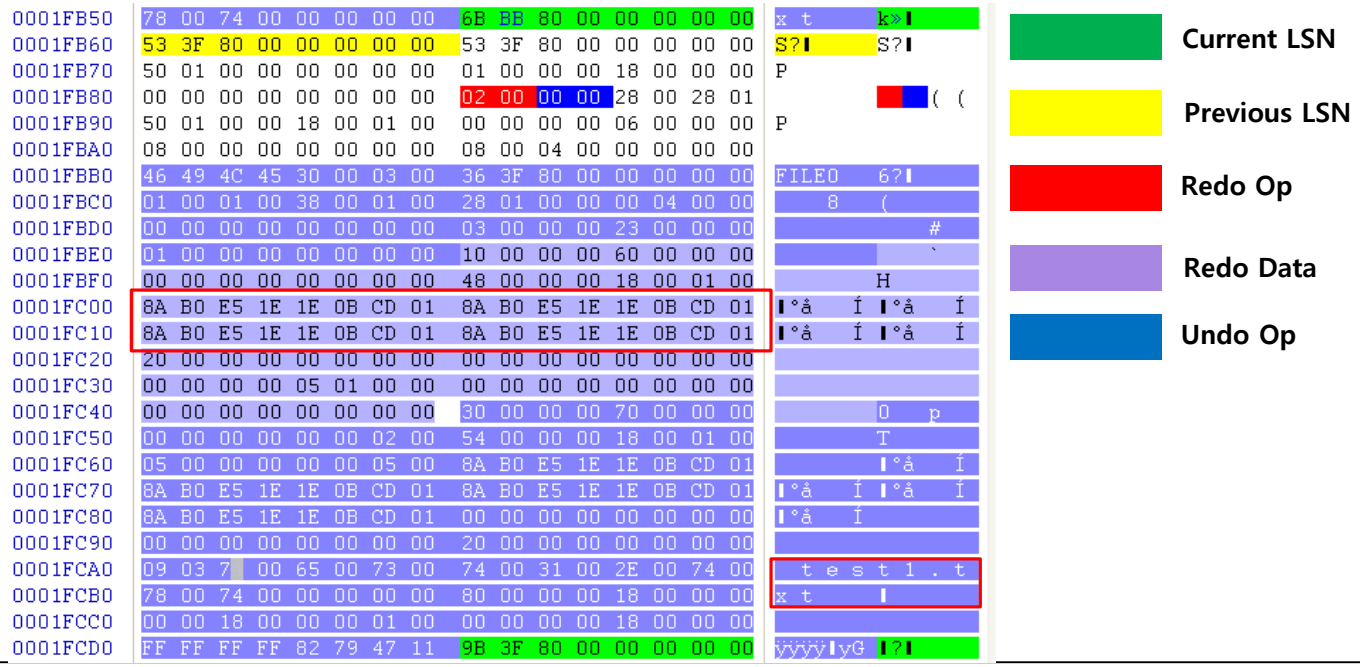
Previous LSN

Redo Op

Undo Op

Creating File/Directory

- **The Information that can be obtained(continued…)**
  - **Creation Time, File Name, Parent Directory Information, File/Directory Separator**
    - ✓ From Redo data of 0x02/0x00(Initialize File Record Segment/Noop) operation record
    - ✓ This Redo data is the content of MFT record.
      - "Creation Time" can be obtained from "$STANDARD_INFORMATION" attribute
      - "File Name" and "Parent Directory Information(Parent File Reference Address)" can be obtained from "$FILE_NAME" attribute.
      - Full path of object(file or directory) can be obtained if "Parent File Reference Address" value is calculated with $MFT
      - If there is "$INDEX_ROOT" attribute, object is directory.



Creating File/Directory

- **Creating a Non-Resident File**
  - The same as that of Resident File
    - ✓ There is no difference in allocating MFT record.
    - ✓ The information that can be obtained is the same as that of creating Resident file.

LSN	Previous LSN	Record Type	Event	Detail	Redo	Undo
8407193	8407176	Update Record			Set Bits In Nonresident Bit Map	Clear Bits In Nonresident Bit Map
8407205	8407193	Update Record			Noop	Deallocate File Record Segment
8407216	8407205	Update Record			Add Index Entry Allocation	Delete Index Entry Allocation
8407240	8407216	Update Record			Initialize File Record Segment	Noop
8407288	8407240	Commit Record			Forget Transaction	Compensation Log Record

Creating File/Directory

- **In case of creating long file name**
  - 0x0E/0x0F(Add Index Entry Allocation/Delete Index Entry Allocation) operation is performed twice. ➔ allocating the Index Entry of long file name

LSN	Previous LSN	Record Type	Event	Detail	Redo	Undo
8403410	0	Update Record			OpenNonresidentAttribute	Noop
8403427	8403410	Update Record			Set Bits In Nonresident Bit Map	Clear Bits In Nonresident Bit Map
8403439	8403427	Update Record			Noop	Deallocate File Record Segment
8403450	8403439	Update Record			OpenNonresidentAttribute	Noop
8403476	8403450	Update Record			Add Index Entry Allocation	Delete Index Entry Allocation
8403503	8403476	Update Record			Add Index Entry Allocation	Delete Index Entry Allocation
8403528	8403503	Update Record			Initialize File Record Segment	Noop
8403594	8403528	Commit Record			Forget Transaction	Compensation Log Record

  - The second $FILE_NAME attribute provides file name created.

```
0001D330  30 00 00 00 78 00 00 00  00 00 00 00 00 00 03 00  0   x
0001D340  5A 00 00 00 18 00 01 00  05 00 00 00 00 00 05 00  Z
0001D350  D8 D4 B8 B2 3D 12 CD 01  D8 D4 B8 B2 3D 12 CD 01  ØÔ¸²=Í ØÔ¸²=Í
0001D360  D8 D4 B8 B2 3D 12 CD 01  D8 D4 B8 B2 3D 12 CD 01  ØÔ¸²=Í ØÔ¸²=Í
0001D370  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0001D380  20 00 00 00 00 00 00 00  0C 02 4C 00 4F 00 4E 00          L O N
0001D390  47 00 5F 00 46 00 7E 00  31 00 2E 00 54 00 58 00  G _ F ~ 1 . T X
0001D3A0  54 00 6D 00 65 00 5F 00  30 00 00 00 88 00 00 00  T m e _ 0   ▌
0001D3B0  00 00 00 00 00 00 02 00  70 00 00 00 18 00 01 00          p
0001D3C0  05 00 00 00 00 00 05 00  D8 D4 B8 B2 3D 12 CD 01          ØÔ¸²=Í
0001D3D0  D8 D4 B8 B2 3D 12 CD 01  D8 D4 B8 B2 3D 12 CD 01  ØÔ¸²=Í ØÔ¸²=Í
0001D3E0  D8 D4 B8 B2 3D 12 CD 01  00 00 00 00 00 00 00 00  ØÔ¸²=Í
0001D3F0  00 00 00 00 00 00 00 00  20 00 00 00 00 00 91 ED              ´í
0001D400  17 01 6C 00 6F 00 6E 00  67 00 5F 00 66 00 69 00   l o n g _ f i
0001D410  6C 00 65 00 5F 00 6E 00  61 00 6D 00 65 00 5F 00  l e _ n a m e _
0001D420  74 00 65 00 73 00 74 00  2E 00 74 00 78 00 74 00  t e s t . t x t
```

Creating File/Directory

- **Obtaining "Create Time" in case of "File System Tunneling"**
  - File System Tunneling ?
    - ✓ When a file is deleted but a new file created with the exact same file name within 15 seconds in the same directory, the previous file's time attributions are assigned to the new file.
  - Operation Modifying "MFT Modified Time" information
    - ✓ Redo : Update Resident Value
    - ✓ Record Offset : 0x38
    - ✓ Attr Offset : 0x20

Redo	Record Offset	Attr Offset	Taget VCN	MFT_Cluster_Index
Add Index Entry Root	0x180	0x110	0x8	0x6
Initialize File Record Segment	0x0	0x0	0x9	0x4
Forget Transaction	0x0	0x0	0x0	0x0
Update Resident Value	0x38	0x20	0x8	0x6

Creating File/Directory

- **Obtaining "Create Time" in case of "File System Tunneling"(Continue…)**
  - Finds modify operation record of 'MFT Modified Time' of the parent directory of a file created
    - ✓ Obtaining "Parent MFT Reference Number"
      - From the redo data of 'Initialize File Record Segment' of a file creation event.
    - ✓ Target VCN = Parent MFT Reference Number / 4
    - ✓ MFT Cluster Index = Parent MFT Reference Number % 4
    - ✓ Find the record of modify operation of 'MFT Modified Time' of a directory that has the same value of the above calculated value of "Target VCN" and "MFT Cluster Index". (Among the older events from the file creation time)
  - Determine the file system tunneling
    - ✓ IF("Create Time" of file != "MFT Modified Time" of parent directory)

      ➔ File System Tunneling!!
    - ✓ This method is not 100% guaranteed because OS creates and deletes dozens of file within 1 second.

Deleting File/Directory

- **Events of Resident File deletion**

8411497	0	Check Point Record	Noop	Noop
8411516	0	Update Record	Delete Index Entry Allocation	Add Index Entry Allocation
8411540	8411516	Update Record	Deallocate File Record Segment	Initialize File Record Segment
8411555	8411540	Update Record	OpenNonresidentAttribute	Noop
8411572	8411555	Update Record	Clear Bits In Nonresident Bit Map	Set Bits In Nonresident Bit Map
8411584	8411572	Commit Record	Forget Transaction	Compensation Log Record

8412302	0	Check Point Record	Noop	Noop
8412321	0	Update Record	Delete Index Entry Allocation	Add Index Entry Allocation
8412345	8412321	Update Record	Deallocate File Record Segment	Initialize File Record Segment
8412360	8412345	Update Record	Clear Bits In Nonresident Bit Map	Set Bits In Nonresident Bit Map
8412372	8412360	Commit Record	Forget Transaction	Compensation Log Record
8412603	0	Check Point Record	Noop	Noop
8412622	0	Update Record	Delete Index Entry Allocation	Add Index Entry Allocation
8412646	8412622	Update Record	Deallocate File Record Segment	Initialize File Record Segment
8412661	8412646	Update Record	Clear Bits In Nonresident Bit Map	Set Bits In Nonresident Bit Map
8412681	8412661	Commit Record	Forget Transaction	Compensation Log Record
8413206	0	Check Point Record	Noop	Noop
8413225	0	Update Record	Delete Index Entry Allocation	Add Index Entry Allocation
8413249	8413225	Update Record	Deallocate File Record Segment	Initialize File Record Segment
8413264	8413249	Update Record	Clear Bits In Nonresident Bit Map	Set Bits In Nonresident Bit Map
8413276	8413264	Commit Record	Forget Transaction	Compensation Log Record

- The record order of Deleting Resident File(Redo/Undo)
  1. 0x0F/0x0E(Delete Index Entry Allocation/Add Index Entry Allocation)
  2. 0x03/0x02(Deallocation File Record Segment/Initialize File Record Segment)
  3. 0x16/0x15(Clear Bits In Nonresident Bit Map/Set Bits In Nonresident Bit Map)
  4. 0x1B/0x01(Forget Transaction/Compensation Log Record)

Deleting File/Directory

- **The Information that can be obtained**
  - **Deleted File Name, Parent Directory Information and File/Directory Separator**
    - ✓ From Undo data of 0x0F/0x0E(Delete Index Entry Allocation/Add Index Entry Allocation) operation record
    - ✓ Undo data is the content of Index Entry($FILE_NAME attribute)
      - Full path of object(file or directory) can be obtained when "Parent File Reference Address" value is calculated with $MFT.
      - "Deleted File Name" can be obtained from "Name" value of $FILE_NAME attribute.
      - "File/Directory Separator" can be obtained from "Flag" value of $FILE_NAME attribute.

```
0002CBE0  7C 59 80 00 00 00 00 00  00 00 00 00 00 00 00 00  |Y
0002CBF0  00 00 00 00 00 00 00 00  90 00 00 00 00 00 8D D3
0002CC00  01 00 00 00 18 00 00 00  00 00 00 00 00 00 00 00
0002CC10  0F 00 0E 00 28 00 00 00  28 00 68 00 44 00 01 00  (   ( h D
0002CC20  00 00 F0 04 00 00 00 00  00 00 00 00 00 00 00 00   ð
0002CC30  2C 00 00 00 00 00 00 00  23 00 00 00 00 00 01 00  ,       #
0002CC40  68 00 54 00 00 00 00 00  05 00 00 00 00 00 05 00  h T
0002CC50  8A B0 E5 1E 1E 0B CD 01  8E 6F 3D 17 56 06 CD 01  °å   Í  o= V Í
0002CC60  5A DF 2A DF 0F 0B CD 01  D3 F9 30 A3 DA 0B CD 01  Zß*ß  Í ÓùO£Ú Í
0002CC70  60 00 00 00 00 00 00 00  5F 00 00 00 00 00 00 00  `       _
0002CC80  20 00 00 00 00 00 00 00  09 03 74 00 65 00 73 00          t e s
0002CC90  74 00 31 00 2E 00 74 00  78 00 74 00 00 00 00 00  t 1 . t x t
```

- Current LSN
- Previous LSN
- Redo Op
- Undo Op
- Undo Data

- **Delete Time**
  - ✓ Get it from "MFT Modified Time" of Parent Directory. (same logic explained in Page 29 and 30.)

Deleting File/Directory

- **In case of deleting a long file name**
  - 0x0F/0x0E(Delete Index Entry Allocation/Add Index Entry Allocation) operation is performed twice. ➔ Deallocating the Index Entry of long file name

LSN	Previous LSN	Record Type	Event	Detail	Redo	Undo
8404406	0	Update Record			Delete Index Entry Allocation	Add Index Entry Allocation
8404431	8404406	Update Record			Delete Index Entry Allocation	Add Index Entry Allocation
8404458	8404431	Update Record			Deallocate File Record Segment	Initialize File Record Segment
8404473	8404458	Update Record			Clear Bits In Nonresident Bit Map	Set Bits In Nonresident Bit Map
8404493	8404473	Commit Record			Forget Transaction	Compensation Log Record

  - For obtaining deleted file name, acquire from second $FILE_NAME attribute

```
0001EDB0  B6 3D 80 00 00 00 00 00  00 00 00 00 00 00 00 00  ¶=▌
0001EDC0  00 00 00 00 00 00 00 00  98 00 00 00 00 00 00 00          ▌
0001EDD0  01 00 00 00 18 00 00 00  00 00 00 00 00 00 00 00
0001EDE0  0F 00 0E 00 28 00 00 00  28 00 70 00 20 01 01 00  ■■(   ( p
0001EDF0  00 00 70 05 00 00 00 00  00 00 00 00 00 00 49 F0    p           Ið
0001EE00  2C 00 00 00 00 00 00 00  23 00 00 00 00 00 01 00  ,       #
0001EE10  70 00 5A 00 00 00 00 00  05 00 00 00 00 00 05 00  p Z
0001EE20  D8 D4 B8 B2 3D 12 CD 01  32 F1 F7 8E 3D 12 CD 01  ØÔ¸²= Í 2ñ÷▌= Í
0001EE30  32 F1 F7 8E 3D 12 CD 01  D8 D4 B8 B2 3D 12 CD 01  2ñ÷▌= Í ØÔ¸²= Í
0001EE40  98 00 00 00 00 00 00 00  91 00 00 00 00 00 00 00  ▌       ´
0001EE50  20 00 00 00 00 00 00 00  0C 02 4C 00 4F 00 4E 00            L O N
0001EE60  47 00 5F 00 46 00 7E 00  31 00 2E 00 54 00 58 00  G _ F ~ 1 . T X
0001EE70  54 00 00 00 00 00 00 00  CF 3D 80 00 00 00 00 00  T       Ï=▌
0001EE80  B6 3D 80 00 00 00 00 00  B6 3D 80 00 00 00 00 00  ¶=▌     ¶=▌
0001EE90  A8 00 00 00 00 00 00 00  01 00 00 00 18 00 00 00  ¨
0001EEA0  00 00 00 00 00 00 00 00  0F 00 0E 00 28 00 00 00          ■■(
0001EEB0  28 00 80 D0 20 01 01 00  00 00 F0 04 00 00 00 00  ( ▌     ð
0001EEC0  00 00 00 00 00 00 00 00  2C 00 00 00 00 00 00 00          ,
0001EED0  23 00 00 00 00 00 01 00  80 00 70 00 00 00 00 00  #       ▌ p
0001EEE0  05 00 00 00 00 00 05 00  D8 D4 B8 B2 3D 12 CD 01          ØÔ¸²= Í
0001EEF0  32 F1 F7 8E 3D 12 CD 01  32 F1 F7 8E 3D 12 CD 01  2ñ÷▌= Í 2ñ÷▌= Í
0001EF00  D8 D4 B8 B2 3D 12 CD 01  98 00 00 00 00 00 00 00  ØÔ¸²= Í ▌
0001EF10  91 00 00 00 00 00 00 00  20 00 00 00 00 00 00 00  ´
0001EF20  17 01 6C 00 6F 00 6E 00  67 00 5F 00 66 00 69 00    l o n g _ f i
0001EF30  6C 00 65 00 5F 00 6E 00  61 00 6D 00 65 00 5F 00  l e _ n a m e _
0001EF40  74 00 65 00 73 00 74 00  2E 00 74 00 78 00 74 00  t e s t . t x t
```

Deleting File/Directory

- **Deleting a Non-Resident File**
  - Same as deletion of Resident File

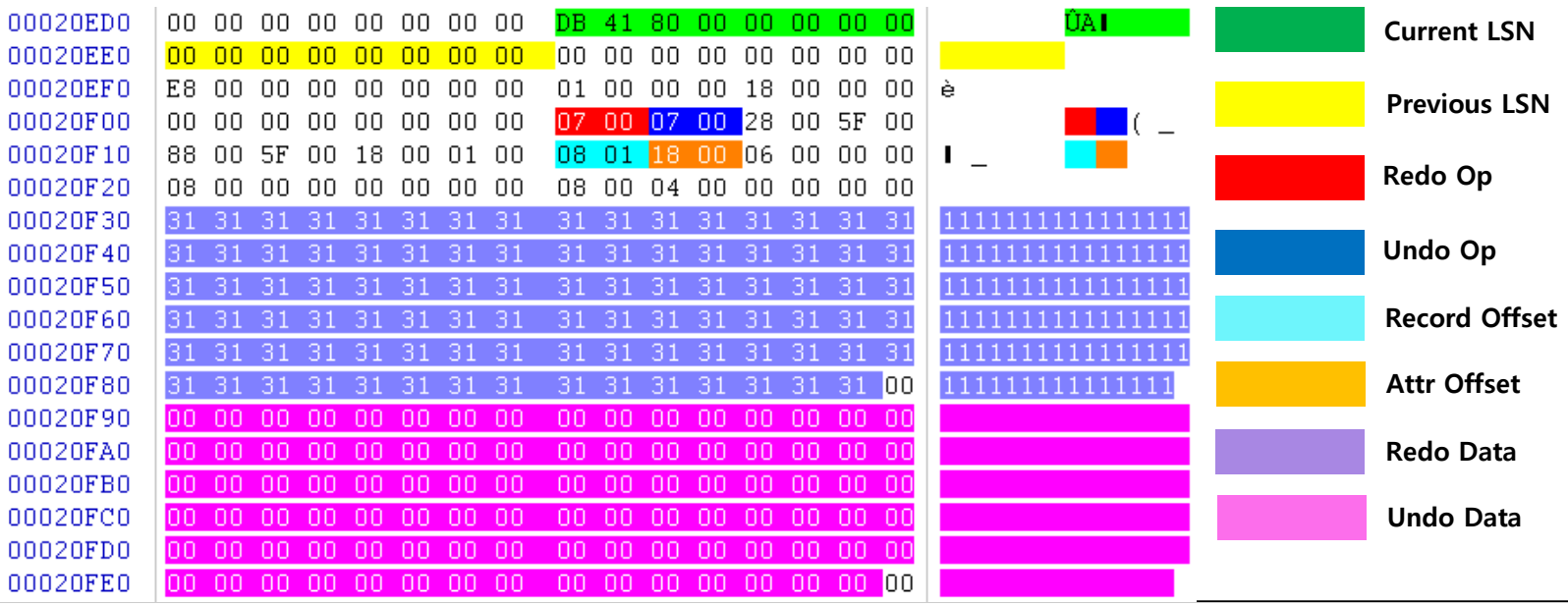
Redo	Undo	sequence nu...	client index	transaction id	Target Attrib...	Record Offset	Attr Offset	Taget LCN
Delete Index Entry Allocation	Add Index Entry Allocation	0x0	0x0	0x18	0xf4	0x0	0x4f0	0x2c
Delete Index Entry Root	Add Index Entry Root	0x0	0x0	0x18	0x18	0x100	0x40	0x40006
Deallocate File Record Segment	Initialize File Record Segment	0x0	0x0	0x18	0x18	0x0	0x0	0x40009
Clear Bits In Nonresident Bit Map	Set Bits In Nonresident Bit Map	0x0	0x0	0x18	0xc8	0x0	0x0	0x3ffff
Forget Transaction	Compensation Log Record	0x0	0x0	0x18	0x18	0x0	0x0	0xde180148
Delete Index Entry Allocation	Add Index Entry Allocation	0x0	0x0	0x18	0xf4	0x0	0x4f0	0x2c
Deallocate File Record Segment	Initialize File Record Segment	0x0	0x0	0x18	0x18	0x0	0x0	0x40009
Clear Bits In Nonresident Bit Map	Set Bits In Nonresident Bit Map	0x0	0x0	0x18	0xc8	0x0	0x0	0x3ffff
Forget Transaction	Compensation Log Record	0x0	0x0	0x18	0x18	0x0	0x0	0xde180148
Delete Index Entry Allocation	Add Index Entry Allocation	0x0	0x0	0x18	0xf4	0x0	0x4f0	0x2c
Delete Index Entry Allocation	Add Index Entry Allocation	0x0	0x0	0x18	0xf4	0x0	0x4f0	0x2c
Deallocate File Record Segment	Initialize File Record Segment	0x0	0x0	0x18	0x18	0x0	0x0	0x40008
Clear Bits In Nonresident Bit Map	Set Bits In Nonresident Bit Map	0x0	0x0	0x18	0xc8	0x0	0x0	0x3ffff
Forget Transaction	Compensation Log Record	0x0	0x0	0x18	0x18	0x0	0x0	0xde180148

- The record order of Non Resident File(Redo/Undo)
  1. 0x0F/0x0E(Delete Index Entry Allocation(or Root)/Add Index Entry Allocation(or Root))
  2. 0x03/0x02(Deallocation File Record Segment/Initialize File Record Segment)
  3. 0x16/0x15(Clear Bits In Nonresident Bit Map/Set Bits In Nonresident Bit Map)
  4. 0x1B/0x01(Forget Transaction/Compensation Log Record)

Writing Data

- **Writing Data of a Resident File(Applicable until Windows XP)**
  - If Redo operation is "Update Resident Value" and "Record Offset" is more than 0xF8 and "Attr Offset" is more than 0x18, it is an operation of updating $DATA attribute
    - ✓ If the length of file name is 1(the short file name), then the start offset of $DATA attribute is 0xF8 within MFT record.
    - ✓ The actual file data starts from 0x18 offset within $DATA attribute
  - If Undo data is all zero, the operation is writing data of new file. If not, modifying data.

Redo	Undo	Target Attrib...	Record Offset	Attr Offset	Taget LCN
Update Resident Value	Update Resident Value	0x18	0x108	0x18	0x40008
Forget Transaction	Compensation Log Record	0x18	0x0	0x0	0xc39c8d20



Writing Data

- **Modifying Data of a Resident File(Applicable until Windows XP)**
  - There are some data left in "Undo Data" area.
    - ✓ Undo Data : the data before modification
    - ✓ Redo Data : the data after modification

Redo	Undo	Target Attrib...	Record Offset	Attr Offset	Taget LCN
Update Resident Value	Update Resident Value	0x18	0x130	0x34	0x40009
Forget Transaction	Compensation Log Record	0x18	0x0	0x0	0xd3ec9164

```
00002950  2A 91 80 00 00 00 00 00  00 00 00 00 00 00 00 00  *'I
00002960  00 00 00 00 00 00 00 00  78 00 00 00 00 00 00 00          x
00002970  01 00 00 00 18 00 00 00  00 00 00 00 00 00 00 00
00002980  07 00 07 00 28 00 21 00  50 00 21 00 18 00 01 00  ( ! P !
00002990  30 01 34 00 00 00 00 00  09 00 00 00 00 00 00 00  0 4
000029A0  09 00 04 00 00 00 00 00  78 78 78 78 78 78 78 78          xxxxxxxx
000029B0  78 78 78 78 78 78 78 78  78 78 78 78 78 78 78 78  xxxxxxxxxxxxxxxx
000029C0  78 78 78 78 78 78 78 78  78 00 00 00 00 00 00 00  xxxxxxxxx
000029D0  61 61 61 61 61 61 61 61  61 61 61 61 61 61 61 61  aaaaaaaaaaaaaaaa
000029E0  61 61 61 61 61 61 61 61  61 61 61 61 61 61 61 61  aaaaaaaaaaaaaaaa
000029F0  61 00 00 00 00 00 00 00  3F 91 80 00 00 00 01 00  a        ?'I
```

- Current LSN
- Previous LSN
- Redo Op
- Undo Op
- Record Offset
- Attr Offset
- Redo Data
- Undo Data

Writing Data

- **Finding Targeted File in Writing Data for a Resident File**
  - Compare values of "Target LCN(VCN)" and "MFT Cluster Index" between "Update Resident Value" and "Initialize File Record Segment" operation records.
  - If the two operation records have the same value of "Target LCN(VCN)" and "MFT Cluster Index", it is considered that data was written to the file via "Initialize File Record Segment" operation.

Redo	Undo	Target Attrib...	Record Offset	Attr Offset	Taget LCN	Cluster Index
Set Bits In Nonresident Bit Map	Clear Bits In Nonresident Bit Map	0xc8	0x0	0x0	0x3ffff	0
Noop	Deallocate File Record Segment	0x18	0x0	0x0	0x40009	4
Add Index Entry Allocation	Delete Index Entry Allocation	0xf4	0x0	0x628	0x2c	0
Initialize File Record Segment	Noop	0x18	0x0	0x0	0x40009	4
Forget Transaction	Compensation Log Record	0x18	0x0	0x0	0x804ea97c	0

Redo	Undo	Target Attrib...	Record Offset	Attr Offset	Taget LCN	Cluster Index
Update Resident Value	Update Resident Value	0x18	0x108	0x18	0x40009	4
Forget Transaction	Compensation Log Record	0x18	0x0	0x0	0xc39c9908	0

Writing Data

- **Writing Data of a Non-Resident File**
  - In case of Non-Resident file, the actual file data is stored in external cluster.
    - ✓ The location information of file data can be obtained from 0x09/0x09(Update Mapping Pairs/Update Mapping Pairs) operation.
    - ✓ If "Attr Offset" is 0x40, the content of Cluster Run can be obtained in Redo data.
      - ➔ In the picture below, the file data is allocated as much as 2 clusters from 38th(0x26) cluster.

```
0002BF40  E8 57 80 00 00 00 00 00  D7 57 80 00 00 00 00 00  èW▌      ×W▌
0002BF50  D7 57 80 00 00 00 00 00  38 00 00 00 00 00 00 00  ×W▌      8
0002BF60  01 00 00 00 18 00 00 00  00 00 00 00 00 00 00 00
0002BF70  09 00 09 00 28 00 08 00  30 00 08 00 18 00 01 00  ■■(    0
0002BF80  30 01 40 00 00 00 00 00  09 00 00 00 00 00 00 00  0 @
0002BF90  09 00 04 00 00 00 00 00  11 02 26 00 00 00 01 00        &
0002BFA0  00 A2 36 A4 5B 07 48 B9  F5 57 80 00 00 00 00 00  ¢6ª[ H¹õW▌
```

- Current LSN
- Previous LSN
- Redo Op
- Undo Op
- Record Offset
- Attr Offset
- Redo Data
- Undo Data

Writing Data

- **Finding a Targeted File in Writing Data for a Non-Resident File**
  - Same as that of Resident File.(Use values of Target LCN and MFT Cluster Index for comparison.)
  - Generally, "Update Mapping Pairs" operation right after file creation event is writing data of Non-Resident file.

Redo	Undo	Target Attrib...	Record Offset	Attr Offset	Taget LCN	Cluster Index
Initialize File Record Segment	Noop	0x18	0x0	0x0	0x40008	6
Forget Transaction	Compensation Log Record	0x18	0x0	0x0	0x804ea97c	0
Delete Attribute	Create Attribute	0x18	0x108	0x0	0x40008	6
Create Attribute	Delete Attribute	0x18	0x108	0x0	0x40008	6
Set Bits In Nonresident Bit Map	Clear Bits In Nonresident Bit Map	0x9c	0x0	0x0	0x3ffdf	0
Set New Attribute Sizes	Set New Attribute Sizes	0x18	0x108	0x0	0x40008	6
Update Mapping Pairs	Update Mapping Pairs	0x18	0x108	0x40	0x40008	6
Set New Attribute Sizes	Set New Attribute Sizes	0x18	0x108	0x0	0x40008	6
Forget Transaction	Compensation Log Record	0x18	0x0	0x0	0x889184b0	0

  - The record order of writing data to Non-Resident File when the file is created.
    1. 0x06/0x05(Delete Attribute/Create Attribute)
    2. 0x05/0x06(Create Attribute/Delete Attribute)
    3. 0x15/0x16(Set Bits In Nonresident Bit Map/Clear Bits In Nonresident Bit Map)
    4. 0x0B/0X0B(Set New Attribute Sizes/ Set New Attribute Sizes)
    5. 0X09/0X09(Update Mapping Pairs/ Update Mapping Pairs)
    6. 0x0B/0X0B(Set New Attribute Sizes/ Set New Attribute Sizes)
    7. 0X1B/0X01(Forget Transaction/Compensation Log Record)

Renaming File/Directory

- **The operations during File/Directory renaming event**
  - Delete and Create of $FILE_NAME attribute
    - ✓ If "Delete Attribute" and "Create Attribute" operations are located next to each other, this is a renaming file event.(These operation's "Record Offset" is 0x98 and "Attr Offset" is 0x00)
      ➔ $FILE_NAME attribute is located at 0x98 offset within MFT record.
    - ✓ The two serial operations should have the same "Target LCN(VCN)" value.

Redo	Undo	Target Attrib...	Record Offset	Attr Offset	Taget LCN
Delete Index Entry Allocation	Add Index Entry Allocation	0xf4	0x0	0x4f0	0x2c
Delete Attribute	Create Attribute	0x18	0x98	0x0	0x40008
Create Attribute	Delete Attribute	0x18	0x98	0x0	0x40008
Add Index Entry Allocation	Delete Index Entry Allocation	0xf4	0x0	0x4f0	0x2c
Forget Transaction	Compensation Log Record	0x18	0x0	0x0	0xa3ef0002

  - The record order of renaming file
    1. 0x0F/0X0E(Delete Index Entry Allocation/Add Index Entry Allocation)
    2. 0x06/0x05(Delete Attribute/Create Attribute)
    3. 0x05/0x06(Create Attribute/Delete Attribute)
    4. 0x0E/0x0F(Add Index Entry Allocation/Delete Index Entry Allocation)
    5. 0x1B/0x01(Forget Transaction/Compensation Log Record)

Renaming File/Directory

- Delete Attribute(0x06) ➔ Create Attribute(0x05)
  - **The File Name before Renaming and after Renaming** can be obtained from each operation's Redo Data($FILE_NAME attribute)
  - **Renaming Time** can be obtained from "MFT Modified Time" of Parent Directory.(same as Page 29 and 30)
  - **File/Directory Separator** can be obtained from "Flag" value of $FILE_NAME attribute.

```
00025D70  AE 4B 80 00 00 00 00 00  96 4B 80 00 00 00 00 00  ®K▌      ▌K▌
00025D80  96 4B 80 00 00 00 00 00  98 00 00 00 00 00 00 00  ▌K▌      ▌
00025D90  01 00 00 00 18 00 00 00  00 00 00 00 00 00 00 00
00025DA0  06 00 05 00 28 00 00 00  28 00 70 00 18 00 01 00  ■■(   ( p
00025DB0  98 00 00 00 06 00 02 00  08 00 00 00 00 00 00 00  ▌■
00025DC0  08 00 04 00 00 00 00 00  30 00 00 00 70 00 00 00  ■      0   p
00025DD0  00 00 00 00 00 00 04 00  56 00 00 00 18 00 01 00          V
00025DE0  05 00 00 00 00 00 05 00  F0 EB 42 97 4C 2E CD 01          ðëB▌L.Í
00025DF0  00 8A 59 AE D5 2D CD 01  12 46 5E 0E 4D 2E 13 60   ▌Y®Õ-Í  F^ M. `
00025E00  F0 EB 42 97 4C 2E CD 01  00 00 00 00 00 00 00 00  ðëB▌L.Í
00025E10  00 00 00 00 00 00 00 00  20 00 00 00 00 00 00 00
00025E20  0A 03 72 00 65 00 6E 00  61 00 6D 00 65 00 2E 00    r e n a m e .
00025E30  74 00 78 00 74 00 00 00  C7 4B 80 00 00 00 00 00  t x t    ÇK▌
00025E40  AE 4B 80 00 00 00 00 00  AE 4B 80 00 00 00 00 00  ®K▌      ®K▌
00025E50  98 00 00 00 00 00 00 00  01 00 00 00 18 00 00 00  ▌
00025E60  00 00 00 00 00 00 00 00  05 00 06 00 28 00 70 00          ■■( p
00025E70  98 00 00 00 18 00 01 00  98 00 00 00 06 00 02 00  ▌        ▌■
00025E80  08 00 00 00 00 00 00 00  08 00 04 00 00 00 00 00          ■
00025E90  30 00 00 00 70 00 00 00  00 00 00 00 00 00 05 00  0   p
00025EA0  58 00 00 00 18 00 01 00  05 00 00 00 00 00 05 00  X
00025EB0  F0 EB 42 97 4C 2E CD 01  00 8A 59 AE D5 2D CD 01  ðëB▌L.Í  ▌Y®Õ-Í
00025EC0  7A D1 52 2B 52 2E CD 01  F0 EB 42 97 4C 2E CD 01  zÑR+R.Í ðëB▌L.Í
00025ED0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00025EE0  20 00 00 00 00 00 00 00  0B 03 72 00 65 00 6E 00            r e n
00025EF0  61 00 6D 00 65 00 32 00  2E 00 74 00 78 00 74 00  a m e 2 . t x t
```

- Current LSN
- Previous LSN
- Redo Op
- Undo Op
- Record Offset
- Attr Offset
- Target LCN
- Redo Data

Moving File/Directory

- The differences between the Move & Rename event
  - File/Directory move event has different parent directory information but the same file name between before and after a file rename.
  - The rest information is same as renaming file event.

$UsnJrnl

- **$UsnJrnl ?**
- The Structure of $UsnJrnl

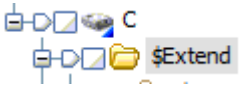
$UsnJrnl ?

- **Journal(Change) Log File of NTFS**
  - This file is used to determine whether any change is occurred in a specific file by applications.
  - From Win7, Journal Function is activated by default
    - ✓ In case of deactivation setting(in Win XP), it is possible to activate through "Fsutil".

      > fsutil usn [createjournal] m=<MaxSize> a=<AllocationDelta> <VolumePath>
    - ✓ For more information about "Fsutil" : http://technet.microsoft.com/en-us/library/cc788042.aspx
  - The file is composed of "$Max" attribute and "$J" attribute
    - ✓ $Max : The meta data of change log is stored.
    - ✓ $J : The actual change log records are stored.
      - Each record has USN(Update Sequence Number) information.
      - The record order is determined with USN.
      - USN = the offset value of a record within $J attribute
      - USN information is also stored in then $STANDARD_INFORMATION attribute of a MFT record

$UsnJrnl ?

- **Journal(Change) Log File of NTFS(continue…)**
  - The file is located under "$Extend" folder.



Name	File Created	Last Written	Entry Modified	Last Accessed	Logical Size
$UsnJrnl:$J					1,246,483,680
$UsnJrnl:$Max					32

  - The size of log data(generally…)
    - ✓ In case of full time use(24 hours/day), the log for 1~2 days are recorded.
    - ✓ In case of regular use(8 hours/day), the log for 4~5 days are recorded.

$UsnJrnl

- $UsnJrnl ?
- **The Structure of $UsnJrnl**

The Structure of $Max attribute

- **The size of $Max attribute**
  - 32 Bytes fixed size
- **The format of $Max attribute**

Offset	Size	Stored Information	Detail
0x00	8	Maximum Size	The maximum size of log data
0x08	8	Allocation Size	The size of allocated area when new log data is saved.
0x10	8	USN ID	The creation time of "$UsnJrnl" file(FILETIME)
0x18	8	Lowest Valid USN	The least value of USN in current records With this value, investigator can approach the start point of first record within "$J" attribute

- **The Structure of c**
  - The log records of variable size are listed consecutively.
  - The zero-filled "Sparse Area" occupies front part of an attribute.

✓ The reason for this structure is because the operating system keeps the same size of the log data saved in the $J attribute.

✓ The record allocation policy of $J attribute

1. The new log records are added at the end of the attribute.
2. If the total size of the added records exceeds "Allocation Size", the operation system assures that the size of the entire log data exceeds "Maximum Size".
3. If the size of the entire log data exceeds "Maximum Size", the front area of attribute is occupied by zero as much as size of "Allocation Size".

✓ Thus, the logical size of $J attribute grow continuously, but the size of area saving actual data is kept constant.

✓ The general size of log data is 0x200000 ~ 0x23FFFF

The Structure of $UsnJrnl

- **The format of record (http://msdn.microsoft.com/en-us/library/aa365722.aspx)**

Offset	Size	Stored Information	Detail
0x00	4	Size of Record	
0x04	2	Major Version	2(Change Journal Software's major version)
0x06	2	Minor Version	0(Change Journal Software's minor version)
0x08	8	MFT Reference Number	"MFT Reference Number" of file or directory that effected by currently change event.
0x10	8	Parent MFT Reference Number	"MFT Reference Number" of parent directory of file and directory that effected by currently change event. The full path information can be obtained with this information and $MFT.
0x18	8	USN	Update Sequence Number
0x20	8	TimeStamp(FILETIME)	Event Time(UTC +0)
0x28	4	Reason Flag	The flag of change event
0x2C	4	Source Information	The subject that triggers change of event
0x30	4	Security ID	
0x34	4	File Attributes	The attribute information of the object effected by current event. Generally, it is used for classifying the object into a file or directory.
0x38	2	Size of Filename	The size of object name effected by current event
0x3A	2	Offset to Filename	The offset of object name within record
0x3C	N	Filename	The object(file or directory) name effected by current event

- The reason for using "Parent MFT Reference Number" instead of "MFT Reference Number"
  - ✓ If "MFT Reference Number" is used, full path information may not be obtained when relevant file is deleted.

The Structure of $UsnJrnl

- **Reason Flag (http://msdn.microsoft.com/en-us/library/aa365722.aspx)**

Flag	Description
0x01	The file was overwritten.
0x02	The file or directory was added to
0x04	The file or directory was truncated.
0x10	The named data streams for a file is overwritten.
0x20	A named data streams for the file were added .
0x40	A named data streams for the file was truncated.
0x100	The file or directory was created for the first time.
0x200	The file or directory was deleted.
0x400	The file's or directory's extended attributes were changed.
0x800	The access rights to the file or directory was changed.
0x1000	The file or directory was renamed.(previous name)
0x2000	The file or directory was renamed.(new name)
0x4000	A user changed the FILE_ATTRIBUTE_NOT_CONTENT_INDEXED attribute.
0x8000	A user has either changed one or more file or directory attributes or one or more time stamps.
0x10000	A hard link was added to or removed from the file or directory
0x20000	The compression state of the file or directory was changed from or to compressed.
0x40000	The file or directory was encrypted or decrypted.
0x80000	The object identifier of the file or directory was changed.
0x100000	The reparse point contained in the file or directory was changed, or a reparse point was added to or deleted from the file or directory.
0x200000	A named stream has been added to or removed from the file, or a named stream has been renamed.
0x80000000	The file or directory was closed.

The Structure of $UsnJrnl

- **Source Information** (http://msdn.microsoft.com/en-us/library/aa365722.aspx)

Flag	Description
0x00	Normal Event
0x01	The operation provides information about a change to the file or directory made by the operating system
0x02	The operation adds a private data stream to a file or directory.
0x04	The operation creates or updates the contents of a replicated file.

The Structure of $UsnJrnl

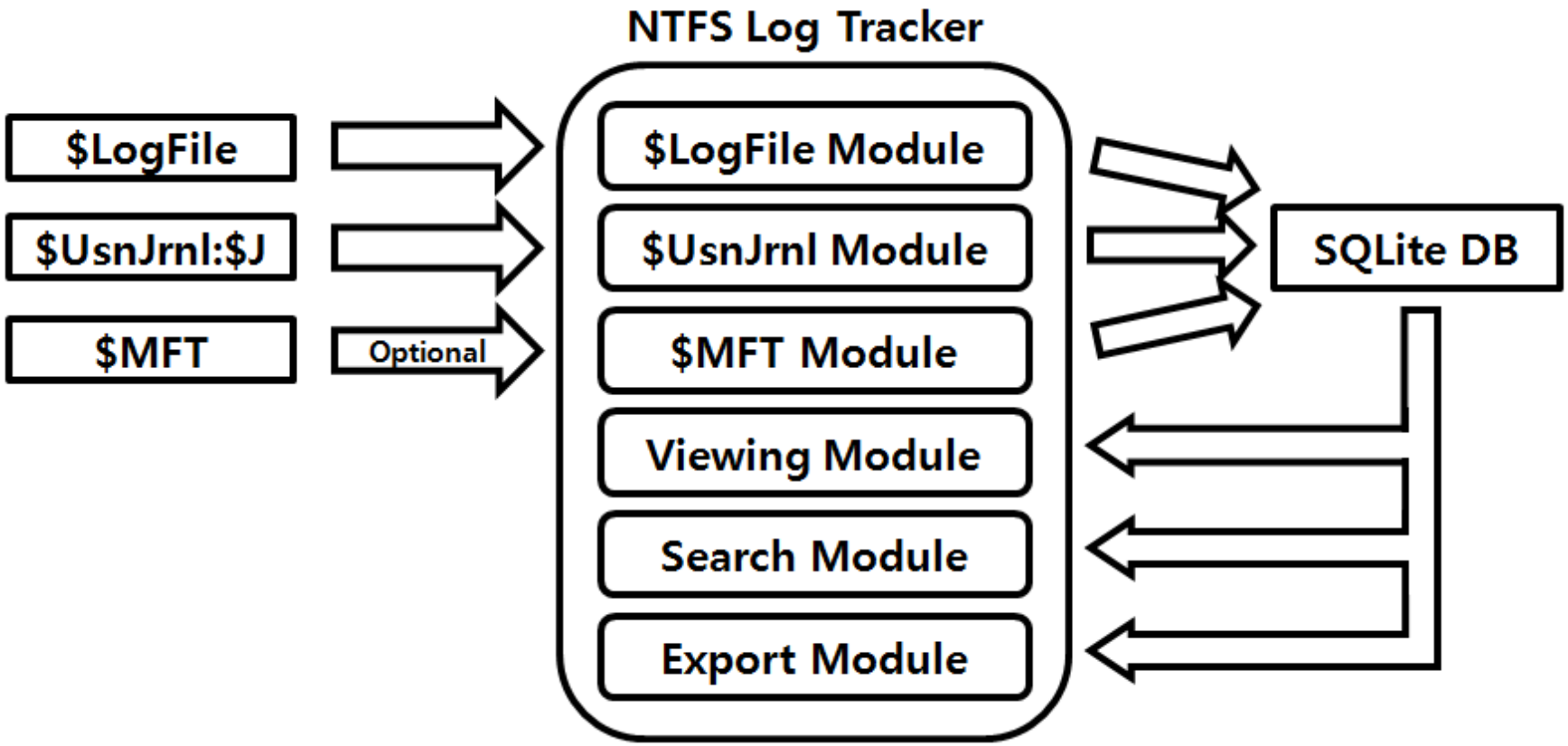
- **File Attribute** (http://msdn.microsoft.com/en-us/library/gg258117.aspx)

Value	Description
0x01	A file that is read-only.
0x02	The file or directory is hidden
0x04	A file or directory that the operating system uses a part of, or uses exclusively.
0x10	The handle that identifies a directory.
0x20	An archive file or directory.
0x40	This value is reserved for system use.
0x80	A file that does not have other attributes set.
0x100	A file that is being used for temporary storage.
0x200	A file that is a sparse file.
0x400	A file or directory that has an associated reparse point, or a file that is a symbolic link.
0x800	A file or directory that is compressed.
0x1000	This attribute indicates that the file data is physically moved to offline storage.
0x2000	The file or directory is not to be indexed by the content indexing service.
0x4000	A file or directory that is encrypted.
0x8000	The directory or user data stream is configured with integrity (only supported on ReFS volumes).
0x10000	This value is reserved for system use.
0x20000	The user data stream not to be read by the background data integrity scanner (AKA scrubber).

NTFS Log Tracker

- **The Tool Design and Development**
- The Toole Functions
- The Comparison of Existing Tools
- Case Study

Tool Design


NTFS Log Tracker
$LogFile
$UsnJrnl:$J
$MFT
Optional
$LogFile Module
$UsnJrnl Module
$MFT Module
Viewing Module
Search Module
Export Module
SQLite DB

The Tool Design and Development

Tool Development : https://code.google.com/p/ntfs-log-tracker/

NTFS Log Tracker v1.1

Parsing File

$LogFile File Path : C:₩Users₩blueangel₩Desktop₩TEST₩$LogFile

$UsnJrnl:$J File Path : C:₩Users₩blueangel₩Desktop₩TEST₩$UsnJrnl:$J

$MFT File Path(Optional) : C:₩Users₩blueangel₩Desktop₩TEST₩$MFT

Parsing

Opening SQLite DB File

SQLite DB File Path :

Open

Filter : Search

CSV Export

$LogFile | $UsnJrnl:$J | $LogFile(Search Result) | $UsnJrnl:$J(Search Result)

< > Page : (1 / 1)

LSN	Event Time	Event	Detail	File Name	Full Path(from $MFT)	Create Time	Modified Time	MFT_Modified Time	Access Time	Redo	Target VCN	Cluster Index
1246242002		Directory Deletion		inst	₩Windows₩SoftwareDistribution₩Download₩4425...	2013-02-27 05:11:39	2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	Deallocate File Record Segment	0x297f	2
1246242353		Writing Content of Non-Resident File	Cluster Number : 3701614(1)							Update Mapping Pairs	0x2970	4
1246242661	2013-02-27 05:11:52	Directory Creation		Install	₩Windows₩SoftwareDistribution₩Download₩Install	2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	Initialize File Record Segment	0x297f	2
1246243003	2013-02-27 05:11:52	File Creation		mpsyschk.exe		2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	Initialize File Record Segment	0x2a20	0
1246243740	2013-02-27 05:11:52	File Creation		BASHV2.DB-journal	₩ProgramData₩Symantec₩Symantec Endpoint Prot...	2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	Initialize File Record Segment	0x4066	6
1246244744				sdmys_B6F210C691D4FA55F5D...	₩ProgramData₩Symantec₩Symantec Endpoint Prot...					Create Attribute	0x4066	6
1246245061		File Deletion		sdmys_B6F210C691D4FA55F5D...	₩ProgramData₩Symantec₩Symantec Endpoint Prot...	2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	Deallocate File Record Segment	0x4066	6
1246245236	2013-02-27 05:11:52	File Creation		BASHV2.DB-journal	₩ProgramData₩Symantec₩Symantec Endpoint Prot...	2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	Initialize File Record Segment	0x4066	6
1246246233				sdmys_B6F210C691D4FA5510D...	₩ProgramData₩Symantec₩Symantec Endpoint Prot...					Create Attribute	0x4066	6
1246246550		File Deletion		sdmys_B6F210C691D4FA5510D...	₩ProgramData₩Symantec₩Symantec Endpoint Prot...	2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	Deallocate File Record Segment	0x4066	6
1246246725	2013-02-27 05:11:52	File Creation		BASHV2.DB-journal	₩ProgramData₩Symantec₩Symantec Endpoint Prot...	2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	Initialize File Record Segment	0x4066	6
1246247746				sdmys_761F41B1A18CF2DA6FC...	₩ProgramData₩Symantec₩Symantec Endpoint Prot...					Create Attribute	0x4066	6
1246248065		File Deletion		sdmys_761F41B1A18CF2DA6FC...	₩ProgramData₩Symantec₩Symantec Endpoint Prot...	2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	Deallocate File Record Segment	0x4066	6
1246248501	2013-02-27 05:11:52	Directory Creation		inst	₩Windows₩SoftwareDistribution₩Download₩718f7...	2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	Initialize File Record Segment	0x4066	6
1246248624				718f72d736c8ecc3d4462ca26b4...	₩Windows₩SoftwareDistribution₩Download₩718f7...	2013-02-15 16:51:26	2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	Update Resident Value	0x2985	6
1246248841	2013-02-27 05:11:52	File Creation		dpx.tmp		2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	Initialize File Record Segment	0x4072	0
1246249408	2013-02-27 05:11:52	File Creation		3d4cd8d3a866df439997400ba5...		2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	Initialize File Record Segment	0x40c2	0
1246249626		Writing Content of Non-Resident File	Cluster Number : 3788936(1)	3d4cd8d3a866df439997400ba5...						Update Mapping Pairs	0x40c2	0
1246250460				update.mum						Create Attribute	0x40c2	0
1246251155	2013-02-27 05:11:52	File Creation		19f7d48b53552d489b9855bd0c...		2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	Initialize File Record Segment	0x40c3	6
1246251365		Writing Content of Non-Resident File	Cluster Number : 3794006(2)	19f7d48b53552d489b9855bd0c...						Update Mapping Pairs	0x40c3	6
1246252154				update.cat						Create Attribute	0x40c3	6
1246252872	2013-02-27 05:11:52	File Creation		30282845_2219809183.xml	₩Windows₩servicing₩Sessions₩30282845_221980...	2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	Initialize File Record Segment	0x40c5	4
1246253632	2013-02-27 05:11:52	File Creation		72566e7fa65aa344b281f7d48d...		2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	Initialize File Record Segment	0x40c5	6
1246253842		Writing Content of Non-Resident File	Cluster Number : 3112128(4)	72566e7fa65aa344b281f7d48d...						Update Mapping Pairs	0x40c5	6
1246254626				x86_microsoft-windows-os-kern...						Create Attribute	0x40c5	6
1246255919	2013-02-27 05:11:52	File Creation		51f94b5a492057489527027da6...		2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	Initialize File Record Segment	0x40c7	2
1246256137		Writing Content of Non-Resident File	Cluster Number : 3702056(4)	51f94b5a492057489527027da6...						Update Mapping Pairs	0x40c7	2
1246256973				x86_microsoft-windows-os-kern...						Create Attribute	0x40c7	2
1246257948	2013-02-27 05:11:52	File Creation		766c6d61d9e72f469763d0976a...		2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	Initialize File Record Segment	0x40cd	6
1246258158		Writing Content of Non-Resident File	Cluster Number : 3702848(4)	766c6d61d9e72f469763d0976a...						Update Mapping Pairs	0x40cd	6
1246258948				x86_microsoft-windows-os-kern...						Create Attribute	0x40cd	6
1246259869	2013-02-27 05:11:52	File Creation		1c53d1ee589bc241972b69b3e3...		2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	Initialize File Record Segment	0x40ce	2
1246260079		Writing Content of Non-Resident File	Cluster Number : 3787700(4)	1c53d1ee589bc241972b69b3e3...						Update Mapping Pairs	0x40ce	2
1246260863				x86_microsoft-windows-os-kern...						Create Attribute	0x40ce	2
1246261859	2013-02-27 05:11:52	File Creation		362fb4eab7e3bf4abc3876f57f3...		2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	2013-02-27 05:11:52	Initialize File Record Segment	0x40d4	4

$LogFile Record Count : 12162 $UsnJrnl Record Count : 1179825

Created by Junghoon Oh(blueangel1275@gmail.com)

NTFS Log Tracker

- The Tool Design and Development
- **The Toole Functions**
- The Comparison of Existing Tools
- Case Study

The Toole Functions

- **Extracting File-Level Event from $LogFile**
  - **Creation/Deletion File/Directory Event(Including "File System Tunneling")**

Event Time	Event	File Name	Full Path(from $MFT)
2013-02-17 13:01:29	File Creation	test3.txt	₩test_directory₩test3.txt
2013-02-17 13:08:39	File Deletion	test3.txt	₩test_directory₩test3.txt
2013-02-17 13:08:41	File Creation(File System Tunneling)	test3.txt	₩test_directory₩test3.txt

    ✓ If there is odd creation event having discontinuous creation time in the middle of events, this event is "File System Tunneling" event.

  - **Writing Data Event**

Event	Detail	File Name
Writing Content of Resident File	Data Offset : 32531576	test.txt
Writing Content of Non-Resident File	Cluster Number : 37575(15)	test2.txt

  - **Renaming/Moving File/Directory Event**

Event	Detail	File Name	Full Path(from $MFT)
Renaming File	test1.txt -> test4.txt	test4.txt	₩test_directory₩test4.txt
Moving Before		test2.txt	₩test_directory₩test2.txt
Moving After		test2.txt	₩test_directory2₩test2.txt

  - In addition, the operation record having same LSN with that of $MFT record is extracted. (including file name)

The Toole Functions

- **Parsing change log from $UsnJrnl**
  - TimeStamp
  - USN
  - FileName
  - Full Path(from $MFT)
  - Event
  - Source Info
  - File Attribute

TimeStamp	USN	FileName	Full Path(from $MFT)	Event	Source Info	File Attribute
2013-01-29 19:51:39	86254360	sce40281.tmp	₩Users₩spejffksem₩AppData₩Local₩Temp₩sce40281.tmp	File_Added, File_Closed	Normal	Archive, Not_Content_Indexed
2013-01-29 19:51:39	86254448	sce40281.tmp	₩Users₩spejffksem₩AppData₩Local₩Temp₩sce40281.tmp	Data_Overwritten	Normal	Archive, Not_Content_Indexed
2013-01-29 19:51:39	86254536	sce40281.tmp	₩Users₩spejffksem₩AppData₩Local₩Temp₩sce40281.tmp	Data_Overwritten, File_Closed	Normal	Archive, Not_Content_Indexed
2013-01-29 19:51:39	86254624	GptTmpl.tmp	₩GptTmpl.tmp	File_Created	Normal	Archive
2013-01-29 19:51:39	86254712	GptTmpl.tmp	₩GptTmpl.tmp	File_Created, File_Closed	Normal	Archive
2013-01-29 19:51:39	86254800	GptTmpl.tmp	₩GptTmpl.tmp	File_Added	Normal	Archive
2013-01-29 19:51:39	86254888	GptTmpl.tmp	₩GptTmpl.tmp	File_Added, Data_Overwritten	Normal	Archive
2013-01-29 19:51:39	86254976	GptTmpl.tmp	₩GptTmpl.tmp	Attr_Changed, File_Added, Data_Overwritten	Normal	Archive
2013-01-29 19:51:39	86255064	GptTmpl.tmp	₩GptTmpl.tmp	Attr_Changed, File_Added, Data_Overwritten, File_Closed	Normal	Archive
2013-01-29 19:51:39	86255152	GptTmpl.inf	₩GptTmpl.inf	File_Truncated	Normal	Archive
2013-01-29 19:51:39	86255240	GptTmpl.inf	₩GptTmpl.inf	File_Added, File_Truncated	Normal	Archive
2013-01-29 19:51:39	86255328	GptTmpl.inf	₩GptTmpl.inf	File_Added, Data_Overwritten, File_Truncated	Normal	Archive
2013-01-29 19:51:39	86255416	GptTmpl.inf	₩GptTmpl.inf	File_Added, Data_Overwritten, File_Truncated, File_Closed	Normal	Archive
2013-01-29 19:51:39	86255504	sce40281.tmp	₩Users₩spejffksem₩AppData₩Local₩Temp₩sce40281.tmp	File_Closed, File_Deleted	Normal	Archive, Not_Content_Indexed
2013-01-29 19:51:39	86255592	GptTmpl.tmp	₩GptTmpl.tmp	File_Closed, File_Deleted	Normal	Archive
2013-01-29 19:51:39	86255680	GPT.INI	₩Windows₩SYSVOL₩domain₩Policies₩{1629D474-A4B9-46F8-AFA7-AE2B49B5CE24}₩GPT.INI	Data_Overwritten	Normal	Archive
2013-01-29 19:51:39	86255760	GPT.INI	₩Windows₩SYSVOL₩domain₩Policies₩{1629D474-A4B9-46F8-AFA7-AE2B49B5CE24}₩GPT.INI	Data_Overwritten, File_Closed	Normal	Archive
2013-01-29 19:51:40	86255840	edb.chk	₩Windows₩security₩database₩edb.chk	Data_Overwritten	Normal	Archive
2013-01-29 19:51:40	86255920	edb.chk	₩Windows₩security₩database₩edb.chk	Data_Overwritten, File_Closed	Normal	Archive

The Toole Functions

- Keyword Search
- Exporting result to CSV file
- Importing SQLite DB(created by NTFS Log Tracker)

NTFS Log Tracker

- The Tool Design and Development
- The Toole Functions
- **The Comparison of Existing Tools**
- Case Study

The Comparison of Existing Tools

- **JP(Windows Journal Parser) :** http://tzworks.net/prototype_page.php?proto_id=5
  - Full Path
    - ✓ JP doesn't support Full Path
  - Renaming File/Directory Event

• Windows Journal Parser

```
02/05/2013, 13:57:03.305, POWERPNT.EXE-E0C7CE05.pf, file_a
02/05/2013, 13:57:30.200, test4567.txt, file_renamed; file
02/05/2013, 13:57:35.551, output.txt, file_deleted; file_c
```

• NTFS Log Tracker

TimeStamp	FileName	Event
2013-02-05 22:57:03	POWERPNT.EXE-E0C7CE05.pf	File_Added, File_Truncate
2013-02-05 22:57:30	test1234.txt	File_Renamed_Old
2013-02-05 22:57:30	test4567.txt	File_Renamed_New
2013-02-05 22:57:30	test4567.txt	File_Renamed_New, File_
2013-02-05 22:57:35	output.txt	File_Closed, File_Deleted

  - Separating File and Directory

• Windows Journal Parser

```
02/05/2013, 14:35:57.664, test_directory, file_created; attrib_changed; file_closed
```

• NTFS Log Tracker

TimeStamp	FileName	Event	File Attribute
2013-02-05 23:35:57	test_directory	File_Created	Directory
2013-02-05 23:35:57	test_directory	File_Created, Attr_Changed	Directory
2013-02-05 23:35:57	test_directory	File_Created, Attr_Changed, File_Closed	Directory

The Comparison of Existing Tools

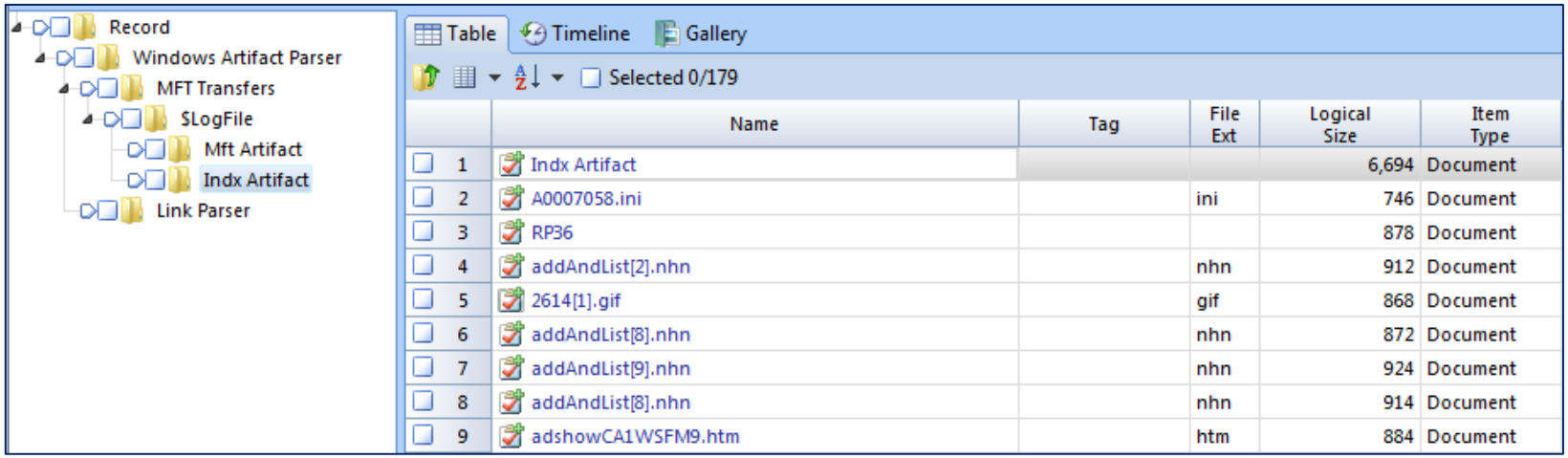
- **$LogFileParser :** https://code.google.com/p/mft2csv/wiki/LogFileParser
  - Parsing record of $LogFile, $UsnJrnl:$J
  - Trace Data Run
  - Not Support Full Path
  - Not for field investigator, for researcher

Offset	MFTReference	MFTBaseRecRef	LSN	LSNPrevious	RedoOperation	UndoOperation	OffsetInMft	FileName
0x00008040	-1		33118228488	33118228458	UpdateNonResidentValue	Noop	640	
0x000080E8	248770		33118228509	33118228488	SetNewAttributeSizes	SetNewAttributeSizes	56	
0x00008180	-1		33118228528	33118228509	ForgetTransaction	CompensationlogRecord	0	
0x000081D8	173507		33118228539	0	SetNewAttributeSizes	SetNewAttributeSizes	56	
0x00008260	173507		33118228556	33118228539	UpdateResidentValue	UpdateResidentValue	56	
0x00008338	173507		33118228583	33118228556	UpdateFileNameAllocation	UpdateFileNameAllocation	0	
0x00008400	248770		33118228608	33118228583	SetNewAttributeSizes	SetNewAttributeSizes	56	
0x00008498	248770		33118228627	33118228608	UpdateNonResidentValue	Noop	720	
0x00008540	248770		33118228648	33118228627	SetNewAttributeSizes	SetNewAttributeSizes	56	
0x000085D8	248770		33118228667	33118228648	UpdateResidentValue	UpdateResidentValue	56	
0x00008640	-1		33118228680	33118228667	ForgetTransaction	CompensationlogRecord	0	

MFTReference	MFTParentReference	USN	Timestamp	Reason	SourceInfo	FileAttributes	FileName	FileNameModified
167894	163478	5771362304	2013-05-14 13:05:11:464:6222	CLOSE+SECURITY_CHANGE	0x00000000	directory	text-base	0
164303	164243	5771362384	2013-05-14 13:05:11:464:6222	SECURITY_CHANGE	0x00000000	directory	prop-base	0
164303	164243	5771362464	2013-05-14 13:05:11:464:6222	CLOSE+SECURITY_CHANGE	0x00000000	directory	prop-base	0
164395	164243	5771362544	2013-05-14 13:05:11:465:6223	SECURITY_CHANGE	0x00000000	directory	props	0
164395	164243	5771362616	2013-05-14 13:05:11:465:6223	CLOSE+SECURITY_CHANGE	0x00000000	directory	props	0
164441	164243	5771362688	2013-05-14 13:05:11:465:6223	SECURITY_CHANGE	0x00000000	directory	text-base	0
164441	164243	5771362768	2013-05-14 13:05:11:465:6223	CLOSE+SECURITY_CHANGE	0x00000000	directory	text-base	0
164243	163478	5771362848	2013-05-14 13:05:11:465:6223	SECURITY_CHANGE	0x00000000	directory	tmp	0
164243	163478	5771362920	2013-05-14 13:05:11:465:6223	CLOSE+SECURITY_CHANGE	0x00000000	directory	tmp	0
163478	127966	5771362992	2013-05-14 13:05:11:465:6223	SECURITY_CHANGE	0x00000000	directory	.svn	0
163478	127966	5771363064	2013-05-14 13:05:11:465:6223	CLOSE+SECURITY_CHANGE	0x00000000	directory	.svn	0
167683	127966	5771363136	2013-05-14 13:05:11:465:6223	SECURITY_CHANGE	0x00000000	archive	index.html	0
167683	127966	5771363216	2013-05-14 13:05:11:465:6223	CLOSE+SECURITY_CHANGE	0x00000000	archive	index.html	0
127966	127734	5771363296	2013-05-14 13:05:11:465:6223	SECURITY_CHANGE	0x00000000	directory	{CD8E1B92-9E0B-4d06-9BE6-	0

The Comparison of Existing Tools

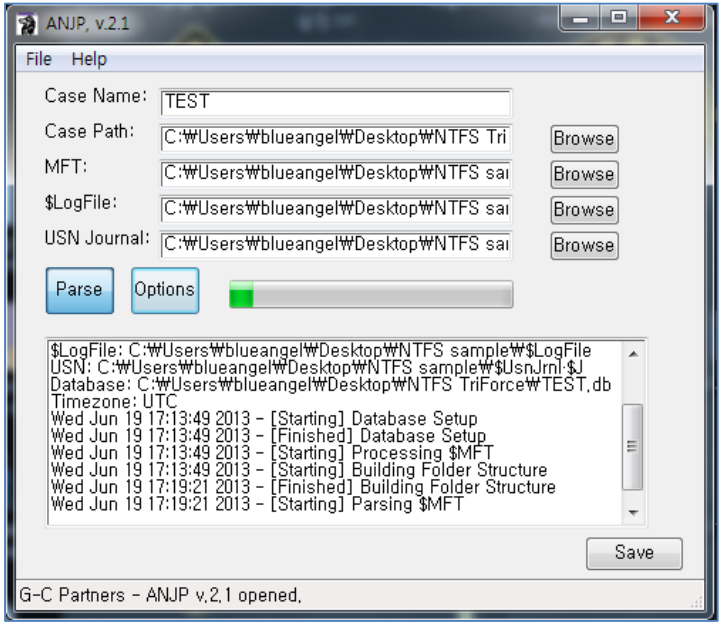
- **Encase v7**
  - MFT Transaction Analysis
    - ✓ Carving MFT Entry, Index Record within $LogFile
    - ✓ Not extracting file-level event



	Name	Tag	File Ext	Logical Size	Item Type
1	Indx Artifact			6,694	Document
2	A0007058.ini		ini	746	Document
3	RP36			878	Document
4	addAndList[2].nhn		nhn	912	Document
5	2614[1].gif		gif	868	Document
6	addAndList[8].nhn		nhn	872	Document
7	addAndList[9].nhn		nhn	924	Document
8	addAndList[8].nhn		nhn	914	Document
9	adshowCA1WSFM9.htm		htm	884	Document

The Comparison of Existing Tools

- **NTFS TriForce(** https://docs.google.com/forms/d/1GzOMe-QHtB12ZnI4ZTjLA06DJP6ZScXngO42ZDGIpR0/viewform)
  - The cross analysis with $MFT, $LogFile, $UsnJrnl
  - Extracting creation, deletion, rename Event
  - Output is SQLite, CSV files



- **X-Ways Forensics**
  - $LogFile Viewer
  - It's commercial tools… I don't use it yet…

NTFS Log Tracker

- The Tool Design and Development
- The Toole Functions
- The Comparison of Existing Tools
- **Case Study**

Case Study 1

- **Extracting malware that is created and then deleted during the boot process**
  - Trace of driver file created during boot process is found.
  - This file trace is not found in $MFT because it is deleted after loading.
  - With "Cluster Number", extracting this driver file from unallocated space.
  - This driver file is confirmed to be malware through reversing.

LSN	Event	Detail	File Name	Full Path(from $MFT)
3447289489	File Creation	Created At 2013-01-25 16:25:33	[redacted].sys	₩Windows₩System32₩drivers₩[redacted].sys
3447289687	Writing Content of Non-Resident File	Cluster Number : 5792454(9)		

Case Study 2

- **Finding traces of malware located only in memory**
  - The malware is located only in memory.
  - This malware detects the system shutdown and drops another malware file.
  - After boot process, this file is loaded to memory and deletes itself
  - The file trace for this malware is not found in $MFT.
  - We find creation and deletion events for this malware file between the shutdown and boot process through cross analysis with $UsnJrnl and Event Log.

TimeStamp	USN	FileName	Full Path(from $MFT)	Event	Source Info	File Attribute
2013-03-07 02:51:15	223271632	[redacted].dll	₩Windows₩System32₩[redacted].dll	File_Added, Data_Overwritten, File_Truncated	Normal	Archive
2013-03-07 02:51:15	223271720	[redacted].dll	₩Windows₩System32₩[redacted].dll	Attr_Changed, File_Added, Data_Overwritten, File_Truncated	Normal	Archive
2013-03-07 02:51:15	223272144	[redacted].dll	₩Windows₩System32₩[redacted].dll	Attr_Changed, File_Added, Data_Overwritten, File_Truncated, File_C...	Normal	Archive
2013-03-07 02:51:16	223277968	[redacted].dll	₩Windows₩System32₩[redacted].dll	File_Closed, File_Deleted	Normal	Archive
2013-03-07 03:03:29	223375360	[redacted].dll	₩Windows₩System32₩[redacted].dll	File_Created	Normal	Archive
2013-03-07 03:03:29	223375448	[redacted].dll	₩Windows₩System32₩[redacted].dll	File_Created, File_Added	Normal	Archive
2013-03-07 03:03:29	223375536	[redacted].dll	₩Windows₩System32₩[redacted].dll	File_Created, File_Added, File_Closed	Normal	Archive

Case Study 3

- **The analysis of $UsnJrnl in Domain Controller(Win2008 R2)**
  - In case of general Win 2008 R2 server, 1~2 days of change log is saved.
  - In case of DC(Domain Controller), more than 1 month of change log is saved.

    ➔ I don't know this reason…
  - It is easy to find the trace of malware for the DC.
    - ✓ Obtained keyword(filename) is used to analyze other systems

$LogFile | $UsnJrnl:$J | $LogFile(Search Result) | $UsnJrnl:$J(Search Result)

< > Page : (1 / 3) Peroid : 2013-01-23 18:27:57 ~ 2013-02-05 13:05:24

TimeStamp	USN	FileName	Full Path(from $MFT)	Event
2013-01-31 06:56:34	91225408	SYMEFA_1.D…	₩System Volume Information₩EfaData₩SYMEFA_1.DB-journal	File_Created, File_Added, Data_Overwritten, File_Closed
2013-01-31 06:56:34	91225512	SYMEFA_1.D…	₩System Volume Information₩EfaData₩SYMEFA_1.DB-journal	File_Renamed_Old
2013-01-31 06:56:34	91225616	sdmys_2A6A…	₩System Volume Information₩EfaData₩sdmys_2A6AC052B827BB65BF79011C	File_Renamed_New
2013-01-31 06:56:34	91225736	sdmys_2A6A…	₩System Volume Information₩EfaData₩sdmys_2A6AC052B827BB65BF79011C	File_Renamed_New, File_Closed, File_Deleted
2013-01-31 06:56:34	91225856	sdmys_7ED74…	₩System Volume Information₩EfaData₩sdmys_7ED7414D8BFDA353C13EDA8A	File_Created, File_Added, Data_Overwritten, File_Closed…
2013-01-31 06:56:34	91225976	sdmys_7ED74…	₩System Volume Information₩EfaData₩sdmys_7ED7414D8BFDA35335930EC8	File_Created, File_Added, Data_Overwritten, File_Closed…
2013-01-31 06:56:42	91226112	BASHV2.DB-j…	₩ProgramData₩Symantec₩Symantec Endpoint Protection₩12.1.1101.401.105₩Data₩BASH…	File_Created
2013-01-31 06:56:42	91226208	BASHV2.DB-j…	₩ProgramData₩Symantec₩Symantec Endpoint Protection₩12.1.1101.401.105₩Data₩BASH…	File_Created, File_Added
2013-01-31 06:56:42	91226304	BASHV2.DB-j…	₩ProgramData₩Symantec₩Symantec Endpoint Protection₩12.1.1101.401.105₩Data₩BASH…	File_Created, File_Added, Data_Overwritten
2013-01-31 06:56:42	91226400	BASHV2.DB-j…	₩ProgramData₩Symantec₩Symantec Endpoint Protection₩12.1.1101.401.105₩Data₩BASH…	File_Created, File_Added, Data_Overwritten, File_Closed
2013-01-31 06:56:42	91226496	BASHV2.DB-j…	₩ProgramData₩Symantec₩Symantec Endpoint Protection₩12.1.1101.401.105₩Data₩BASH…	File_Renamed_Old

Case Study 4

- **CTF** (thanks to **Deok9~**)
  - 2013 CodeGate CTF, Forensic 200
  - Analyzing the $LogFile from given disk image.
    - ✓ The creation event under specific path is found.

	µTorrent.lnk	₩Users₩Public₩Desktop₩µTorrent.lnk
	Desktop.ini	₩Users₩CodeGate_Forensic₩AppData₩Roaming₩Microsoft₩Windows₩Start Menu₩Programs₩Accessc
	dht.dat	₩Users₩Administrator₩AppData₩Roaming₩uTorrent₩dht.dat
	resume.dat	₩Users₩Administrator₩AppData₩Roaming₩uTorrent₩resume.dat
	rss.dat	₩Users₩Administrator₩AppData₩Roaming₩uTorrent₩rss.dat
	settings.dat	₩Users₩Administrator₩AppData₩Roaming₩uTorrent₩settings.dat
	settings.dat.old	₩Users₩Administrator₩AppData₩Roaming₩uTorrent₩settings.dat.old
	10E6FBE4D921B475FA5FEC6E9A535A540D6FEED1	₩Users₩Administrator₩AppData₩Roaming₩uTorrent₩dlimagecache₩10E6FBE4D921B475FA5FEC6E9A5
	utorrent.lng	₩Users₩Administrator₩AppData₩Roaming₩uTorrent₩utorrent.lng
	3609FC884502A1DF0AA5D9D160C827BB1BD51FC	₩Users₩Administrator₩AppData₩Roaming₩uTorrent₩apps₩3609FC884502A1DF0AA5D9D160C827BB1E
	featuredContent.btapp	₩Users₩Administrator₩AppData₩Roaming₩uTorrent₩apps₩featuredContent.btapp
	plus.btapp	₩Users₩Administrator₩AppData₩Roaming₩uTorrent₩apps₩plus.btapp
	player.btapp	₩Users₩Administrator₩AppData₩Roaming₩uTorrent₩apps₩player.btapp
	2D78C93EC367E6C1D9894103FA04B3BE5B20A84E	₩Users₩Administrator₩AppData₩Roaming₩uTorrent₩dlimagecache₩2D78C93EC367E6C1D9894103FAC
	whatsnew-ut.btapp	₩Users₩Administrator₩AppData₩Roaming₩uTorrent₩apps₩whatsnew-ut.btapp
	83DD5C860D7C31A1D3588629CA65A66BEA75689	₩Users₩Administrator₩AppData₩Roaming₩uTorrent₩dlimagecache₩83DD5C860D7C31A1D3588629CA
	32F529521A3DEC709F97F761F192AABF29BDC408	₩Users₩Administrator₩AppData₩Roaming₩uTorrent₩dlimagecache₩32F529521A3DEC709F97F761F192
	BBEEC0395D21A2A7F91889D7C7509F3D5D46FC05	₩Users₩Administrator₩AppData₩Roaming₩uTorrent₩dlimagecache₩BBEEC0395D21A2A7F91889D7C75
	98E3ED7A3B1D58C3E51BAAFC15A3D9876B4396B	₩Users₩Administrator₩AppData₩Roaming₩uTorrent₩dlimagecache₩98E3ED7A3B1D58C3E51BAAFC15
	C5BED7C03B5061C637102B5BB2299385699ABDD	₩Users₩Administrator₩AppData₩Roaming₩uTorrent₩dlimagecache₩C5BED7C03B5061C637102B5BB22
File Creation	052b585f1808716e1d12eb55aa646fc4984bc862	₩Users₩CodeGate_Forensic₩AppData₩Roaming₩Microsoft₩Windows₩Start Menu₩Programs₩Startup
	Startup	₩Users₩CodeGate_Forensic₩AppData₩Roaming₩Microsoft₩Windows₩Start Menu₩Programs₩Startup
	052b585f1808716e1d12eb55aa646fc4984bc862	₩Users₩CodeGate_Forensic₩AppData₩Roaming₩Microsoft₩Windows₩Start Menu₩Programs₩Startup

    - ✓ All file-level events reveal how the CTF challenge were created.

  - Detail Solution

    ➔ http://forensicinsight.org/wp-content/uploads/2013/03/F-INSIGHT-CodeGate-2013-Write-ups.pdf

Conclusion

Conclusion

- **NTFS's log file** : $LogFile, $UsnJrnl
- **Analysis that rely on the $MFT only are limited and miss the following.**
  - The trace of delete file
  - The repeated event to specific file
- **It's necessary to analyze file system event with $LogFile, $UsnJrnl.**
- **What NTFS Log Tracker has over other tools**
  - The analysis of $LogFile, $UsnJrnl
  - Supporting Full Path information(with $MFT)
  - Keyword search, Exporting result to CSV file, Importing SQLite file(created by this tool)

Question and Answer