NTFS Log Tracker

blueangel

blueangel1275@gmail.com

forensic-note.blogspot.kr

Junghoon Oh

Index

Introduction

- **NTFS 의 로그 파일**
  - $LogFile : 트랙젝션 로그
  - $UsnJrnl : 변경 로그

- 기존의 NTFS에 대한 파일 시스템 포렌식
  - $MFT 파일 중심의 파일 시스템 이벤트 분석
    - ✓ $MFT : 파일 시스템에 존재하는 모든 파일, 디렉터리의 메타데이터를 저장하는 파일
  - 삭제 파일의 경우, $MFT 에 메타데이터가 남아있지 않을 가능성이 있음
    - ✓ 삭제된 파일에 대한 흔적 추적의 어려움
      - 시스템 드라이브(EX : C:₩)의 경우, 운영체제가 항상 임시 파일을 생성
      - Win7의 부터 주기적인 가비지 컬렉션
      - SSD의 경우, TRIM 작업에 의해 비할당영역 정리

- **$LogFile, $UsnJrnl 분석**
  - 특정 기간 동안 일어난 파일 시스템 이벤트를 분석 가능
  - $MFT 에 남아있지 않은 이벤트 분석 가능
    - ✓ 삭제된 파일에 대한 히스토리
    - ✓ 특정 파일에 대한 히스토리($MFT 의 경우, 마지막 접근/쓰기 시간만 기록됨)
      - 각 접근 시간 파악
      - 각 쓰기 시간 파악

$LogFile

- **$LogFile ?**
- $LogFile 구조
- $LogFile 이벤트 분석

$LogFile ?

NTFS 트랜젝션 로그 파일

- 시스템 오류나 갑작스런 전원 차단 발생시, 작업 중이던 파일 복구를 위해 사용

- 모든 트랜젝션 작업을 레코드 단위로 기록
  - ✓ 새로운 파일/디렉토리 생성
  - ✓ 파일/디렉토리 삭제
  - ✓ 파일/디렉토리 내용 변경
  - ✓ MFT 엔트리 내용 변경

- 각 작업 레코드는 고유의 LSN($LogFile Sequence Number)을 가짐
  - ✓ 순차적으로 증가

- 복구를 위해 각 레코드는 작업 데이터와 작업 전 데이터를 가짐
  - ✓ Redo : 작업한 데이터
  - ✓ Undo : 작업 전 데이터

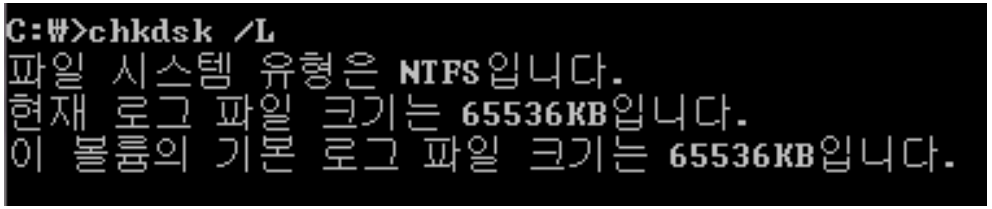
- 각 볼륨마다 하나씩 존재

- MFT 엔트리 번호 2에 위치

Entry 번호	Entry 이름	설명
0	$MFT	NTFS 상의 모든 파일들의 MFT Entry 정보
1	$MFTMirr	$MFT 파일의 일부 백업본
2	$LogFile	메타데이터의 트랜잭션 저널 정보
3	$Volume	볼륨의 레이블, 식별자, 버전 등의 정보
4	$AttrDef	속성의 식별자, 이름, 크기 등의 정보
5	.	볼륨의 루트 디렉터리

$LogFile ?

- **$LogFile 크기**
  - 일반적인 하드디스크 볼륨에서는 64M 크기
  - 볼륨 용량에 따라 크기가 달라질 수 있지만 기본적으로는 최대 64M 이하임
  - 64M 기준, 일반적인 컴퓨터 활동(웹서핑, 문서 작업...)을 할 경우, 2~3 시간 정도의 로그가 남음
  - 포렌식 준비도 측면에서 저장 용량을 늘릴 필요가 있음
- **크기 조절**
  - chkdsk 명령의 /L 옵션에 따라 크기 조절 가능
  - "/L : 파일크기(KB 단위)" 형식으로 지정
  - 크기가 지정 되지 않으면 현재 크기 표시

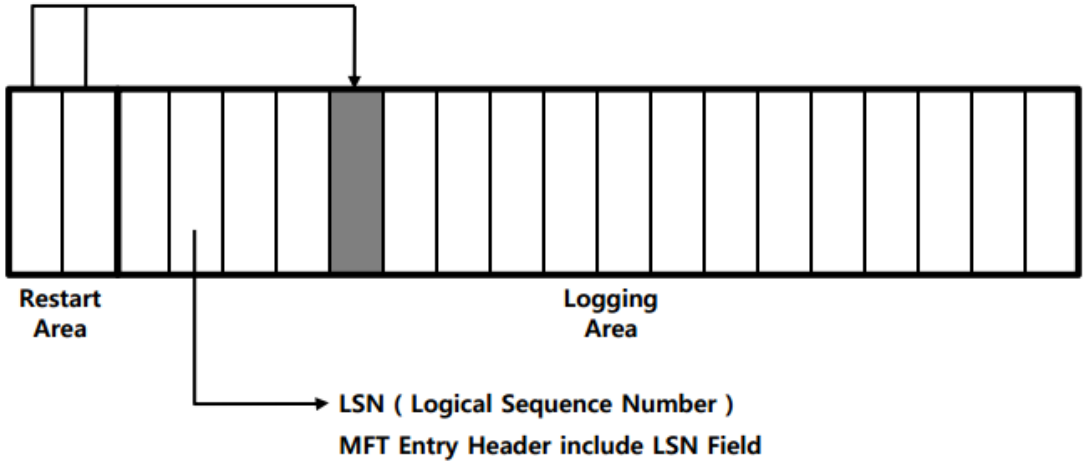


$LogFile

- $LogFile ?
- **$LogFile 구조**
- $LogFile 이벤트 분석

전체 구조

- **재시작 영역(Restart Area)와 로깅 영역(Logging Area)로 나누어짐**
  - 각 영역의 구성단위는 페이지(크기 : 0x1000)
  - 재시작 영역
    - ✓ 가장 마지막(현재 작업 중인) 작업 레코드를 가리킴
    - ✓ 파일의 첫 두 페이지 영역(0x0000~0x2000)
  - 로깅 영역
    - ✓ 실제 작업 레코드들이 기록됨
    - ✓ 재시작 영역 바로 다음부터 시작(0x2000~)
    - ✓ 버퍼 페이지 영역과 일반 페이지 영역으로 나누어짐



재시작 영역 구조

- **가장 마지막(현재 작업 중인) 작업 레코드를 가리킴**
  - Current LSN 정보를 통해 가장 마지막 작업 레코드의 LSN 번호를 알 수 있음

- **연속된 두 페이지로 구성, 두 번째 페이지는 백업용**
  - 각 페이지는 매직넘버(RSTR)로 시작됨

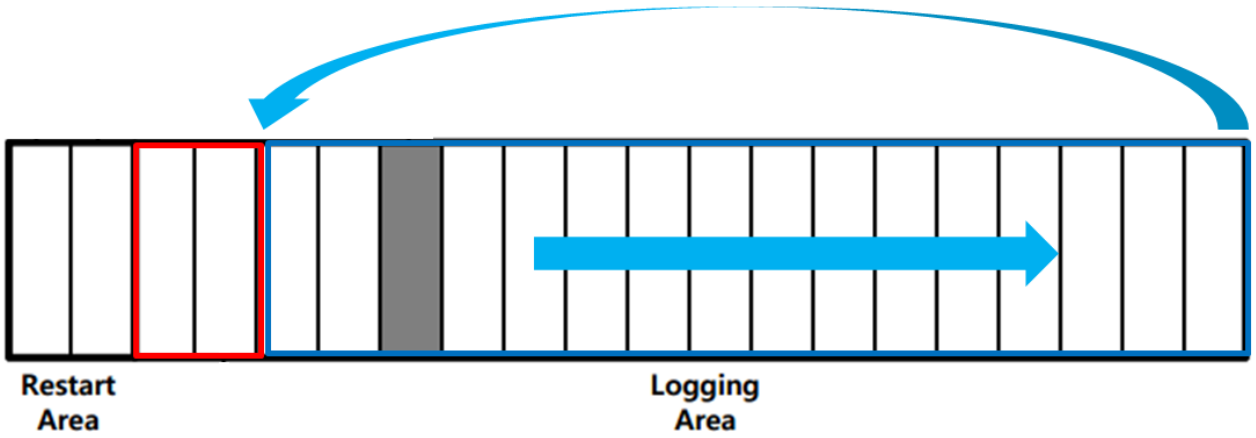
- **재시작 영역 헤더 포멧**

<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</th><th>E F</th></tr>
<tr><td>“RSTR”
(Magic Number)</td><td>Update Sequence Offset</td><td>Update Sequence Count</td><td colspan="4">Check Disk LSN</td></tr>
<tr><td colspan="2">System Page Size</td><td colspan="2">Log Page Size</td><td>Restart Offset</td><td>Minor Version</td><td>Major Version</td></tr>
<tr><td colspan="7">Update Sequence Array</td></tr>
<tr><td colspan="4">Current LSN</td><td>Log Client</td><td>Client List</td><td>Flags</td></tr>
</table>

로깅 영역 구조

- **실제 작업 레코드들이 기록됨**

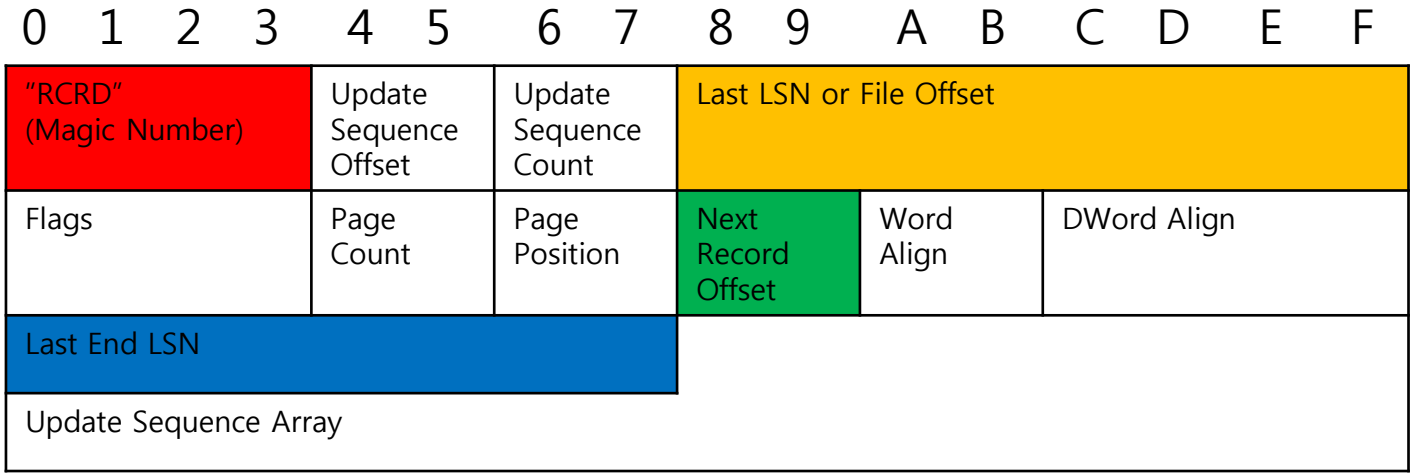
- **버퍼 페이지 영역과 일반 페이지 영역으로 나누어짐**
  - 버퍼 페이지 영역 ➔ 첫 두 페이지(0x2000~0x4000)
    ➔ 순차적으로 레코드가 기록됨
    ➔ 페이지가 레코드로 꽉 차면 페이지 내용을 일반 페이지 영역에 기록
    ➔ 최근 작업 레코드들은 버퍼 페이지 영역에 존재
  - 일반 페이지 영역 ➔ 버퍼 페이지 영역을 제외한 나머지 영역(0x4000~)



페이지 구조

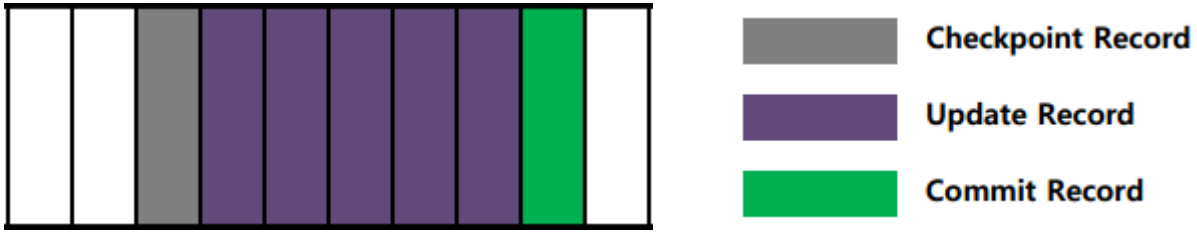
- 페이지 구성
  - 하나의 헤더와 다수의 작업 레코드들로 구성됨
  - 마지막 레코드가 페이지를 넘어가면 다음 페이지에 이어서 기록됨

- 페이지 헤더 : 페이지의 메타 데이터가 저장됨
  - Magic Number : "RCRD"
  - Last LSN : 페이지를 넘어가는 레코드를 포함해서 가장 큰 LSN
  - Next Record Offset : Last LSN에 해당하는 레코드의 페이지 내 Offset
  - Last End LSN : 페이지를 넘어가지 않는 레코드들 중에 가장 큰 LSN



작업 레코드 구조

- **작업 레코드**
  - 실제 트랜젝션 작업의 내용이 기록됨
  - 여러 작업 레코드가 순차적으로 모여서 하나의 트랜젝션 작업을 이룸
    - ✓ Check Point Record : 트랜젝션 시작 레코드
    - ✓ Update Record : 중간 작업 레코드
    - ✓ Commit Record : 트랜젝션 마지막 레코드

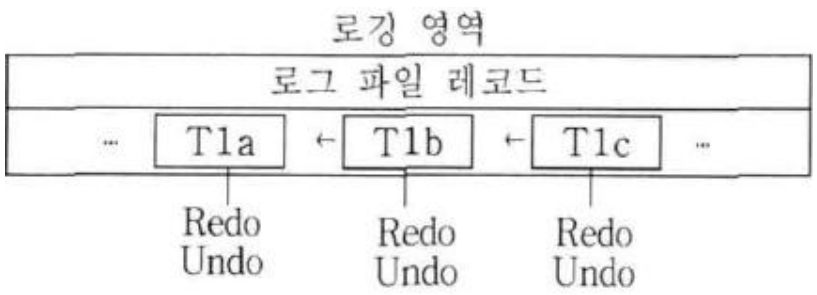


  - Check Point Record 외 레코드들은 자신의 이전 작업 레코드의 LSN을 가지고 있음

작업 레코드 구조

- **작업 레코드(계속)**
  - 작업 레코드 구성 : 레코드 헤더와 데이터로 구성 됨
    - ✓ 레코드 헤더 : 레코드 메타 데이터 저장, 고정 크기(0x58)
    - ✓ 레코드 데이터
      - Redo : 작업 후 내용(예 : 쓰기 작업이면 쓰여진 데이터)
      - Undo : 작업 전 내용(예 : 쓰기 작업이면 쓰여지기 전 데이터)

  - **에러 복구시의 작업 내용**
    - ✓ Commit Record 부터 이전 LSN 정보를 이용, 역으로 추적하면서 Undo 데이터 적용



작업 레코드 구조

- **작업 레코드 헤더 포멧**
  - This LSN : 현재 작업 레코드의 LSN
  - Previous LSN : 이전 작업 레코드의 LSN
  - Client Undo LSN : 복구 시, 다음 Undo 작업을 가지고 있는 레코드의 LSN, 보통 Previous LSN과 동일
  - Client Data Length : 레코드의 크기, Redo Op 시작 위치부터 이 값을 더하면 레코드 끝을 구할 수 있음
  - Record Type : 0x02 (Check Point Record), 0x01(그 외 Record)
  - Flags : 0x01(현재 레코드가 페이지를 넘어감), 0x00(현재 레코드가 페이지를 넘어가지 않음)

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

작업 레코드 구조

- **작업 레코드 헤더 포멧(계속)**
  - Redo Op : Redo 연산 코드
  - Undo Op : Undo 연산 코드
  - Redo Offset : Redo 데이터 시작 Offset(Redo Op 위치부터)
  - Redo Length : Redo 데이터 길이
  - Undo Offset : Undo 데이터 시작 Offset(Redo Op 위치부터)
  - Undo Length : Undo 데이터 길이

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											

작업 레코드 구조

▪ 작업 레코드 헤더 포멧(계속)

- LCNs to Follows : 0x01(이어지는 레코드가 있음), 0x00(이어지는 레코드가 없음)
- Record Offset
  - ✓ MFT 레코드에 대한 작업일 경우, Redo/Undo 데이터가 적용되는 속성의 MFT 레코드 내 Offset
  - ✓ MFT 레코드에 대한 작업이 아닌 경우, 값은 0x00
- Attr Offset
  - ✓ MFT 레코드에 대한 작업일 경우, Redo/Undo 데이터가 적용되는 속성 내 Offset
  - ✓ MFT 레코드에 대한 작업이 아닌 경우, Redo/Undo 데이터가 적용되는 클러스터 내 Offset
- MFT Cluster Index : MFT 엔트리가 있는 하나의 클러스터 내에서 몇 번째 엔트리에 해당하는지에 대한 값
  - ✓ 1번째(0x0000), 2번째(0x0002), 3번째(0x0003), 4번째(0x0006)

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											

작업 레코드 구조

- **작업 레코드 헤더 포멧(계속)**
  - Target VCN : Redo/Undo 데이터가 적용되는 $MFT 상의 VCN(Virtual Cluster Number)
  - Target LCN : Redo/Undo 데이터가 적용되는 디스크 상의 LCN(Logical Cluster Number)

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											

작업 레코드 구조

- **Redo/Undo 연산 코드**

NTFS 작업	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

작업 레코드 구조

- **Redo/Undo 연산 코드(계속)**

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 ?
- $LogFile 구조
- **$LogFile 파일 단위 이벤트 분석**

$LogFile 파일 단위 이벤트 분석

- **$LogFile 파일 단위 이벤트 분석의 필요성**
  - $LogFile 의 각 작업레코드에 저장된 정보는 파일 단위의 이벤트가 아님
    - ✓ 여러 작업 레코드들이 모여서 하나의 트랜젝션 이벤트를 이룸
  - 분석가에게 의미 있는 파일 단위의 이벤트로 변경해야 함~!!!
  - 분석 대상 이벤트
    - ✓ 파일 생성
    - ✓ 파일 삭제
    - ✓ 파일 데이터 작성
    - ✓ 파일명 변경
    - ✓ 파일 이동

파일 생성 이벤트

- **Resident File 생성 관련 이벤트**

LSN	Previous LSN	Record Type	Event	Detail	File 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

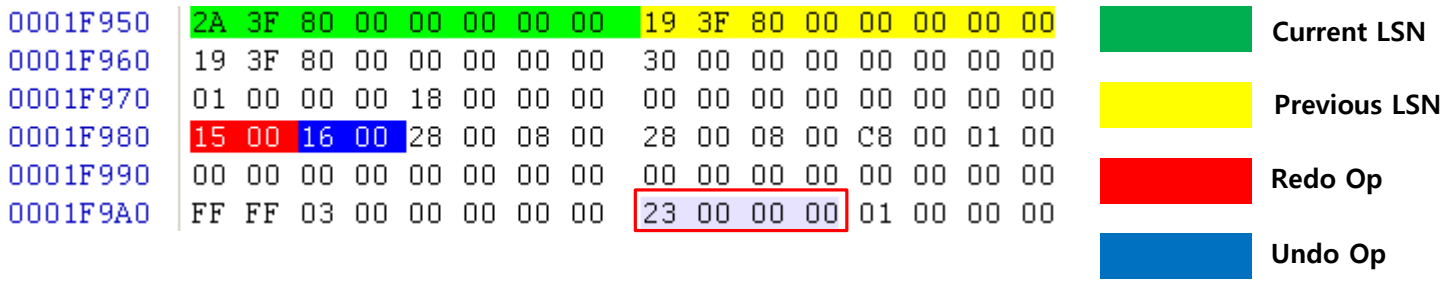
LSN	Previous LSN	Record Type	Event	Detail	File Name	Full Path	Redo	Undo
8405882	0	Check Point Record					Noop	Noop
8405901	0	Update Record					Set Bits In Nonresident Bit Map	Clear Bits In Nonresident Bit Map
8405913	8405901	Update Record					Noop	Deallocate File Record Segment
8405924	8405913	Update 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	File 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

- Resident 파일 생성 이벤트 순서(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)
- 위 화면은 $LogFile 작업 레코드들을 그대로 파싱해주는 Research Version 도구의 캡쳐 화면임

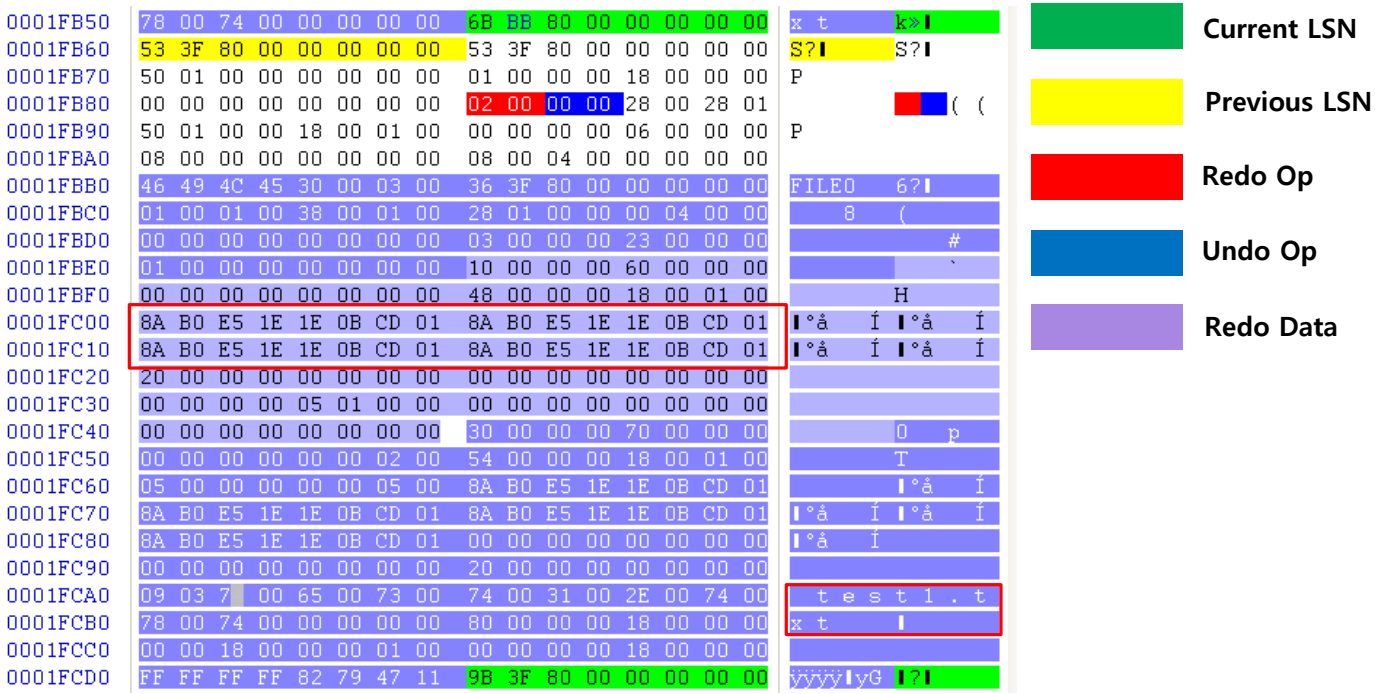
파일 생성 이벤트

- **Resident File 생성 관련 이벤트에서 얻어 올 수 있는 정보 1**
  - MFT 레코드 번호, 생성 파일 전체 경로
    - ✓ 0x15/0x16(Set Bits In Nonresident Bit Map/Clear Bits In Nonresident Bit Map) 작업의 Redo 데이터에서 얻어옴
    - ✓ Redo 데이터의 첫 4바이트는 작업 대상 MFT 레코드 번호임
    - ✓ MFT 레코드 번호를 통해 해당 파일의 정보를 가져올 수 있음
      - 해당 MFT 레코드의 $FILE_NAME 속성에서 생성 파일명 획득
      - MFT 번호를 알면 MFT 해석을 통해 생성된 파일의 전체 경로를 가져 올 수 있음



파일 생성 이벤트

- **Resident File 생성 관련 이벤트에서 얻어 올 수 있는 정보 2**
  - **파일 생성 시간**과 **생성 파일명**, **부모디렉터리 정보, 파일/디렉터리 구분**
    - ✓ 0x02/0x00(Initialize File Record Segment/Noop) 작업의 Redo 데이터에서 얻어옴
    - ✓ Redo 데이터 내용은 MFT 레코드의 내용
      - $STANDARD_INFORMATION 속성에서 파일 생성 시간을 가져옴
      - $FILE_NAME 속성에서 생성 파일의 이름을 가져옴, Parent File Reference Address 값을 통해 부모 디렉토리와 전체 경로(with $MFT)를 알 수 있음
      - $INDEX_ROOT 속성이 있다면 디렉토리 생성임



파일 생성 이벤트

- **Non Resident 파일 생성 이벤트**
  - Resident 파일과 동일
    - ✓ MFT 레코드 할당하는 것에서는 Resident 파일 생성 작업과 차이 없음
    - ✓ Resident 파일 생성 경우와 동일하게 정보 획득 가능

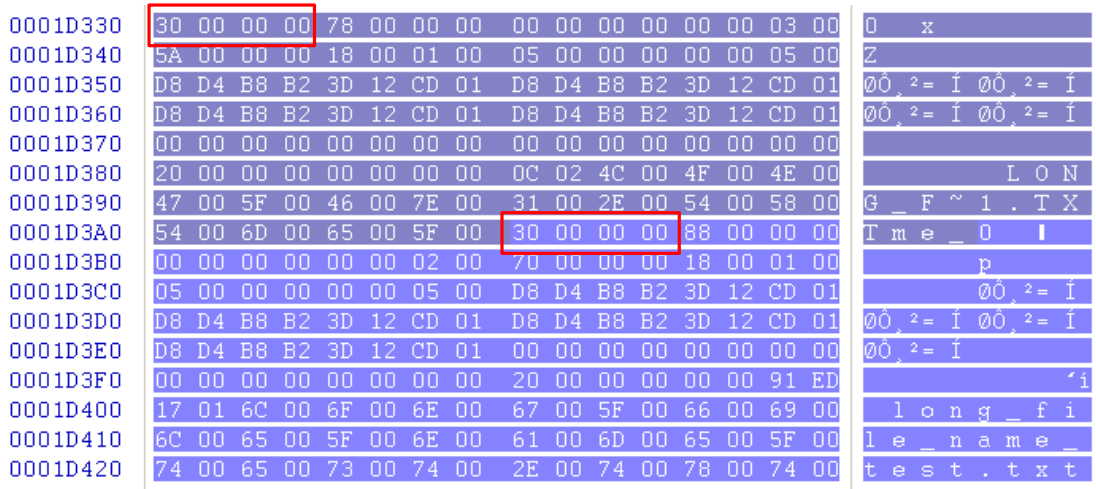
LSN	Previous LSN	Record Type	Event	Detail	Redo	Undo
8407193	8407176	Update Record			Set Bits In Nonresident Bit Map	Clear Bits In Nonresident Bit Map
8407205	8407193	Update Record			Noop	Deallocate File Record Segment
8407216	8407205	Update Record			Add Index Entry Allocation	Delete Index Entry Allocation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

파일 생성 이벤트

- **긴 파일명의 파일 생성일 경우**
  - 0x0E/0x0F(Add Index Entry Allocation/Delete Index Entry Allocation) 작업을 한 번 더 반복함
    ➔ 긴 파일명이기 때문에 Index Entry를 하나 더 할당

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

  - 파일명을 가져올 경우, 두 번째 $FILE_NAME 속성에서 가져옴



파일 생성 이벤트

- **파일 시스템 터널링에 대한 생성 시간 획득**
  - 파일 시스템 터널링 ?
    - ✓ 동일한 디렉터리 아래에서 파일이 삭제되고 15초 안에 동일한 이름의 파일이 생성되면 이전에 존재했던 파일의 시간정보가 새로 생성된 파일에 그대로 저장됨
  - 디렉터리의 MFT Modified Time 수정하는 작업 레코드
    - ✓ 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

파일 생성 이벤트

- **파일 시스템 터널링에 대한 생성 시간 획득(계속)**
  - 생성 파일의 부모 디렉터리 MFT Modified Time 수정 이벤트 찾기
    - ✓ Parent MFT Reference Number 획득
      - 파일 생성 이벤트의 "Initialize File Record Segment" Redo 데이터에서 획득
    - ✓ Target VCN = Parent MFT Reference Number / 4
    - ✓ MFT Cluster Index = Parent MFT Reference Number % 4
    - ✓ 계산한 Target VCN, MFT Cluster Index 값을 가진 디렉터리의 MFT Modified Time 수정하는 작업 레코드를 찾음.(생성 이벤트를 기준으로 이 전 이벤트들 중에서)
  - 판단 기준
    - ✓ IF(파일의 생성 시간 != 부모 디렉터리의 MFT Modified Time 수정 시간)

      ➔ 파일 시스템 터널링 이벤트~!!
    - ✓ 100% 다 찾을 수 있는 것이 아님, OS가 1초에 수십씩 파일을 생성하고 삭제하기 때문에…

      ➔ 생성 이벤트들 중, 생성시간이 연속적이지 않은 이벤트의 경우, 파일 시스템 터널링으로 판단

파일 삭제 이벤트

- **Resident File 삭제 관련 이벤트**

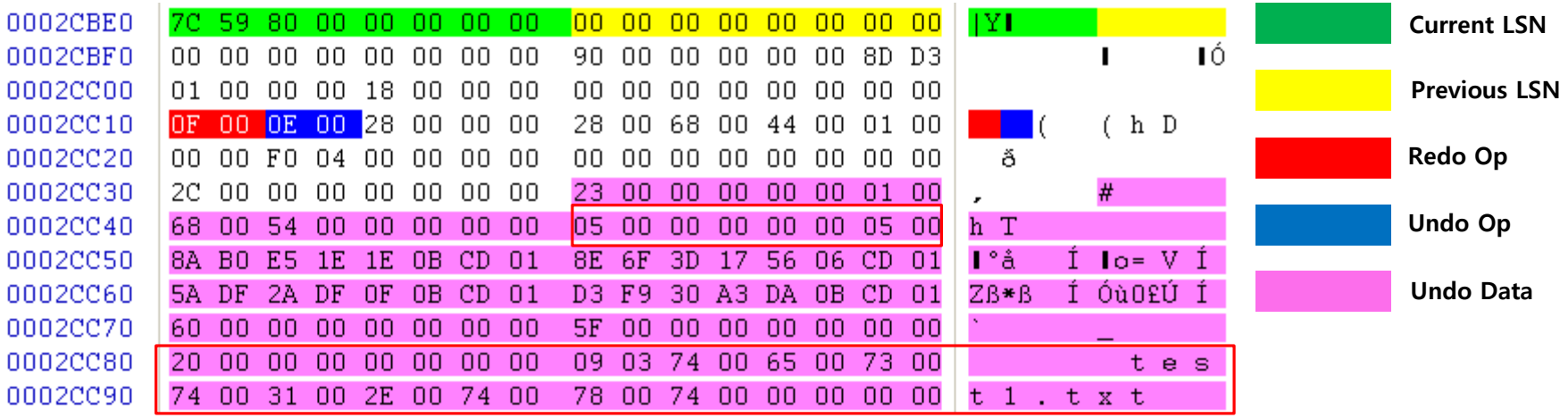
8411497	0	Check Point Record	Noop	Noop
8411516	0	Update Record	Delete Index Entry Allocation	Add Index Entry Allocation
8411540	8411516	Update Record	Deallocate File Record Segment	Initialize File Record Segment
8411555	8411540	Update Record	OpenNonresidentAttribute	Noop
8411572	8411555	Update Record	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

- Resident File 삭제 이벤트 순서(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)

파일 삭제 이벤트

- **Resident File 삭제 관련 이벤트에서 얻어 올 수 있는 정보**
  - **삭제된 파일명과 부모 디렉터리 정보 그리고 파일/디렉터리 구분**
    - ✓ 0x0F/0x0E(Delete Index Entry Allocation/Add Index Entry Allocation) 작업의 Undo 데이터에서 얻어옴
    - ✓ Undo 데이터의 내용은 Index Entry 안의 Content 내용($FILE_NAME 속성)
      - Parent File Reference Address 값을 통해 부모 디렉토리와 전체 경로를 알 수 있음(with $MFT)
      - Name 값을 통해 삭제된 파일명 획득
      - Flag 정보를 통해 파일 or 디렉터리 구분



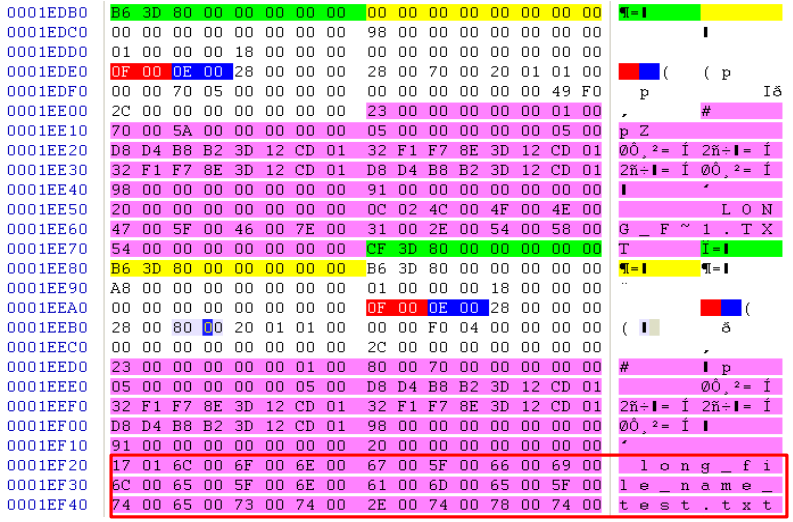
- **삭제 시간** : 부모 디렉터리의 MFT Modified 시간 정보로 부터 획득

파일 삭제 이벤트

- **긴 파일명의 파일을 삭제할 경우**
  - 0x0F/0x0E(Delete Index Entry Allocation/Add Index Entry Allocation) 작업이 두 번 일어남
    ➔ 긴 파일명이기 때문에 Index Entry가 두 개이기 때문

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

  - 삭제된 파일명을 가져오기 위해서는 두 번째 0x0F/0x0E(Delete Index Entry Allocation/Add Index Entry Allocation) 작업의 Undo 데이터에서 가져옴



파일 삭제 이벤트

- **Non Resident 파일 삭제 이벤트**
  - Resident 삭제 작업과 동일하게 판단
    - ✓ Resident 삭제 작업과 마찬가지로 파일명이 긴 경우, Delete Index Entry Allocation 작업이 두 번 일어남
    - ✓ 삭제 파일명, 전체 경로는 Resident 파일 삭제의 경우와 동일하게 획득

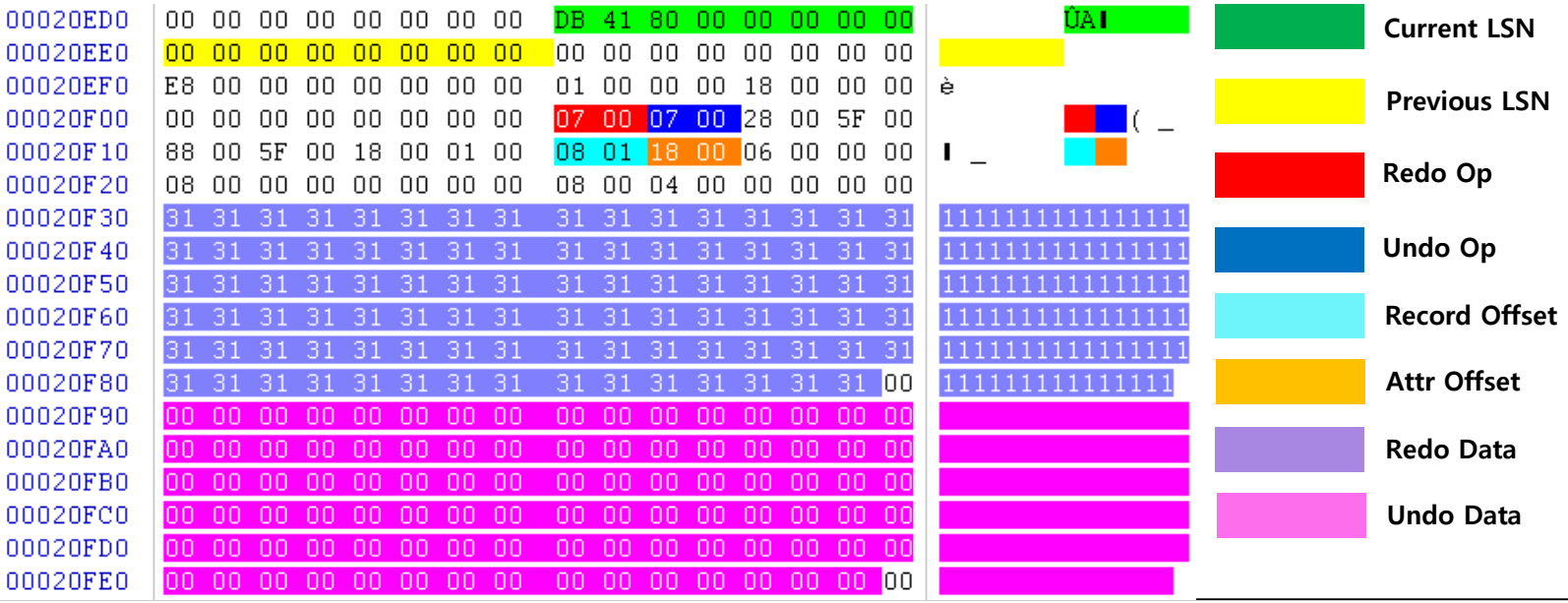
Redo	Undo	sequence nu...	client index	transaction id	Target Attrib...	Record Offset	Attr Offset	Taget LCN
Delete Index Entry Allocation	Add Index Entry Allocation	0x0	0x0	0x18	0xf4	0x0	0x4f0	0x2c
Delete Index Entry Root	Add Index Entry Root	0x0	0x0	0x18	0x18	0x100	0x40	0x40006
Deallocate File Record 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

- Non Resident File 삭제 이벤트 순서(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)

파일 데이터 작성 이벤트

- **Resident File 파일 데이터 작성(Windows XP 까지 적용됨, Win7 부터 적용 안됨)**
  - Redo 작업이 Update Resident Value 이고 Record Offset 이 0xF8 이상, 그리고 Attr Offset 이 0x18 이상이면 $DATA 속성에 대한 업데이트 작업이라고 볼 수 있음
    - ✓ 파일명 길이가 1인 경우(짧은 파일명), $Data속성의 시작위치는 0xF8
    - ✓ $DATA 속성에서 0x18 위치부터 실제 데이터가 들어감
  - Undo의 데이터가 모두 0이면 새로운 파일 내용 작성, 그렇지 않으면 파일 내용 수정

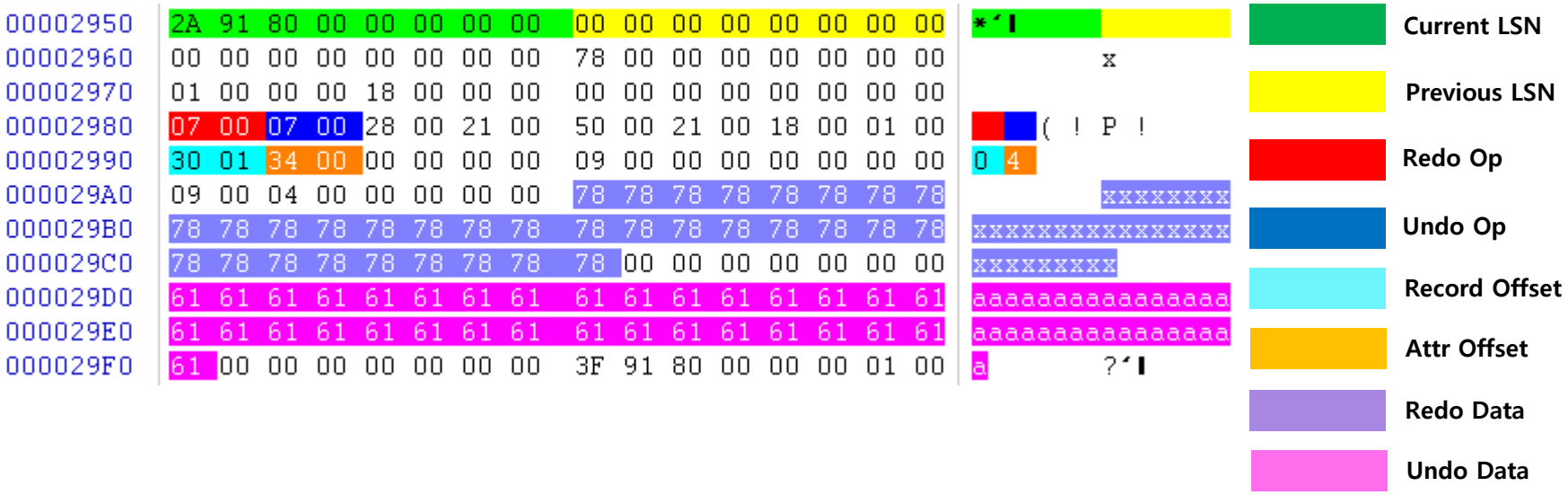
Redo	Undo	Target Attrib...	Record Offset	Attr Offset	Taget LCN
Update Resident Value	Update Resident Value	0x18	0x108	0x18	0x40008
Forget Transaction	Compensation Log Record	0x18	0x0	0x0	0xc39c8d20



파일 데이터 작성 이벤트

- **Resident File 파일 데이터 수정(Windows XP 까지 적용됨, Win7 부터 적용 안됨)**
  - Undo 에 데이터가 있음
    - ✓ Undo의 데이터가 수정 전의 내용
    - ✓ Redo의 데이터가 수정 후의 내용

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



파일 데이터 작성 이벤트

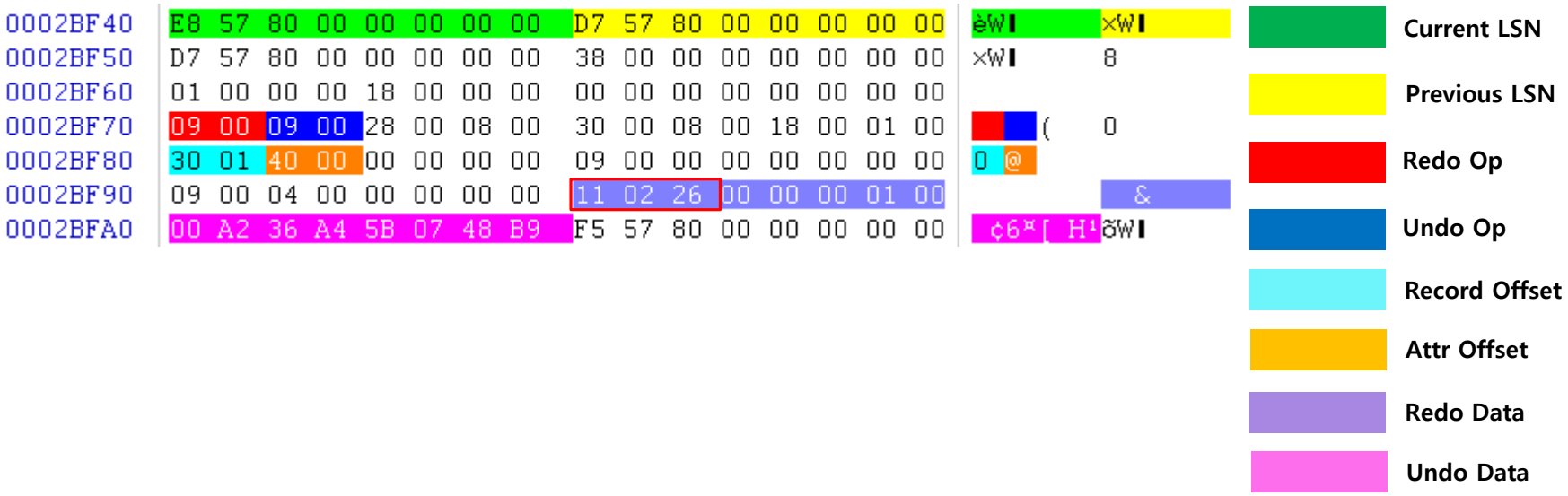
- **대상 파일 찾기**
  - Update Resident Value 작업의 Target LCN, MFT Cluster Index 값과 Initialize File Record Segment 작업의 Target LCN(VCN), MFT Cluster Index 값을 비교
  - 같은 Target LCN(VCN), MFT Cluster Index 값을 가지고 있으면 Initialize File Record Segment 작업을 통해 생성된 파일의 내용을 작성/수정한 것이라 볼 수 있음
  - 이 방법도 100% 모두 찾는 것이 아님(OS가 1초에 수십 개씩 파일을 지우고 삭제하기 때문)

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

파일 데이터 작성 이벤트

- **Non Resident 내용/작성 수정 이벤트**
  - Non Resident 파일의 경우, 실제 파일의 내용이 외부 클러스터에 저장됨
    - ✓ 0x09/0x09(Update Mapping Pairs/Update Mapping Pairs) 작업을 통해 데이터 작성 위치를 확인할 수 있음
    - ✓ Attr Offset이 0x40 일 경우, Cluster Run 작성 내용을 Redo 데이터에서 획득할 수 있음(0x41일 경우, 확인 불가)
      - ➔ 아래의 경우, 0x26번째 클러스터부터 2클러스터가 사용되었음



파일 데이터 작성 이벤트

- **Non Resident 파일 생성시, 해당 파일의 데이터 위치 파악하기**
  - Resident 파일 내용 작성의 경우와 마찬가지로 Target LCN, MFT Cluster Index 비교를 통해 데이터가 작성되는 파일을 찾을 수 있음
  - 일반적으로 파일 생성 이벤트 다음에 바로 오는 Update Mapping Pairs 작업이 생성한 파일의 데이터 쓰기 작업임

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

- Non Resident 파일 생성시, 데이터 작성 이벤트
  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)

파일명 변경 이벤트

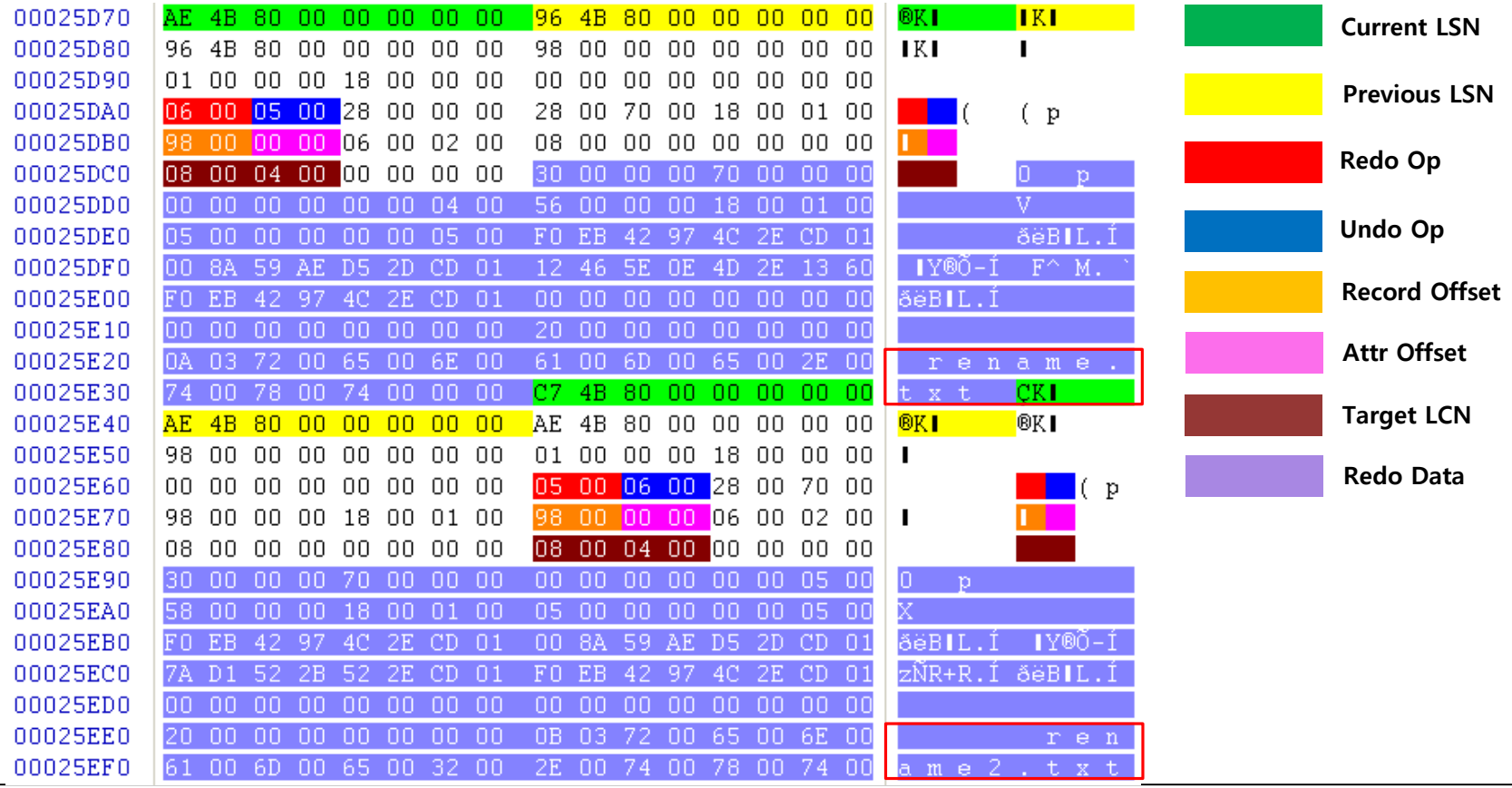
- **파일명 변경 시, 일어나는 작업**
  - $FILE_NAME 속성 삭제, 추가 작업
    - ✓ Record Offset 이 0x98, Attr Offset 이 0x00 인 Delete Attribute와 Create Attribute 작업이 연속적으로 오면 파일명 변경 ➔ 일반적으로 $FILE_NAME 속성은 MFT 레코드에서 0x98 위치에 있음
    - ✓ 두 작업의 Target LCN이 동일해야 함

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

  - 파일명 변경 이벤트 순서
    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)

파일명 변경 이벤트

- Delete Attribute(0x06) ➔ Create Attribute(0x05)
  - 각 작업의 Redo Data($FILE_NAME 속성) 에서 **변경 전 파일명**과 **변경 후 파일명**을 알 수 있음
  - **파일명 변경 시간** : 부모 디렉터리의 MFT Modified 시간 정보로 부터 획득
  - Flag 값을 통해 이름을 변경한 객체가 파일인지 디렉터리인지 구분



파일 이동 이벤트

- 파일명 변경 이벤트와 차이점
  - 변경 전과 변경 후의 이름이 같고 부모 디렉터리 정보가 다르다면 **이동 이벤트**라고 볼 수 있음
  - 나머지 정보들은 파일명 변경 이벤트와 동일

$UsnJrnl

- **$UsnJrnl ?**
- $UsnJrnl 구조

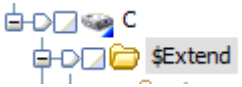
$UsnJrnl ?

- **NTFS 변경 로그 파일**
  - 응용 프로그램이 특정 파일의 변경 여부를 파악하기 위해 사용
  - 기본적으로 Windows 7 부터 활성화되어 있음
    - ✓ 비활성화 되어있을 시, Fsutil 로 활성화 시킬 수 있음

      > fsutil usn [createjournal] m=<MaxSize> a=<AllocationDelta> <VolumePath>
    - ✓ Fsutil 의 자세한 사용법은 http://technet.microsoft.com/en-us/library/cc788042.aspx
  - $Max 속성과 $J 속성으로 구성
    - ✓ $Max : 변경 로그의 기본 메타 데이터 저장
    - ✓ $J 속성 : 실제 변경 로그 레코드 저장
      - 각 레코드들은 USN(Update Sequence Number) 정보를 가짐
      - USN 정보를 통해 각 레코드들의 순서 구분
      - 실제 USN 값은 $J 속성 내에서의 레코드의 Offset 값
      - USN 값은 MFT 엔트리의 $STANDARD_INFORMATION 속성에도 저장되어 있음

$UsnJrnl ?

- **NTFS 변경 로그 파일(계속)**
  - 루트에 있는 "$Extend" 폴더 아래 위치



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

  - 기록 되는 로그 데이터의 양(일반적으로...)
    - ✓ 컴퓨터를 계속 사용할 경우, 1~2일 정도의 로그가 남음
    - ✓ 규칙적으로 쓸 경우(하루 8시간), 4~5일 정도의 로그가 남음

$UsnJrnl

- $UsnJrnl ?
- **$UsnJrnl 구조**

$Max 속성의 구조

- **$Max 속성의 크기**
  - 32 Byte 고정 크기를 가짐
- **$Max 속성의 저장 정보**

Offset	Size	Stored Information	Detail
0x00	8	Maximum Size	로그 데이터의 최대 크기
0x08	8	Allocation Size	새로운 데이터가 저장될 때 할당 되는 영역의 크기
0x10	8	USN ID	"$UsnJrnl" 파일의 생성시간(FILETIME)
0x18	8	Lowest Valid USN	현재 저장된 레코드 중 가장 작은 USN 값 이 정보를 통해 $J 속성 내 첫 번째 레코드로 바로 이동 가능

$UsnJrnl 구조

- **$J 속성 구조**
  - 가변 크기의 로그 레코드들이 연속적으로 나열됨
  - 속성의 앞 부분은 0으로 채워진 "Sparse Area" 를 가짐

✓ 이러한 구조를 가지는 이유는 운영체제가 $J 속성에 저장되는 로그 데이터의 크기를 일정하게 유지하려고 하기 때문임

✓ $J 속성의 레코드 할당 정책

1. 새로운 로그 레코드들은 속성 끝에 추가됨
2. 추가된 레코드들의 총 크기가 "Allocation Size"를 넘으면 추가 레코드들을 포함하여 전체 로그 데이터의 크기가 "Maximum Size" 를 넘는지 확인
3. 전체 로그 데이터의 크기가 "Maximum Size" 를 넘는 다면 로그 데이터의 앞 부분을 "Allocation Size" 만큼 0으로 채워 "Sparse Area" 로 만듬

✓ 따라서 $J 속성의 논리적인 크기는 계속 커지지만 실제 데이터가 할당된 영역은 일정하게 유지됨

✓ 일반적으로 0x200000 ~ 0x23FFFFF 의 로그 데이터를 저장

$UsnJrnl 구조

- **$J 속성의 로그 레코드 구조(http://msdn.microsoft.com/en-us/library/aa365722.aspx)**

Offset	Size	Stored Information	Detail
0x00	4	Size of Record	레코드 크기
0x04	2	Major Version	2(현재 일반적으로 사용되는 Change Journal Software의 버전은 2.0)
0x06	2	Minor Version	0(현재 일반적으로 사용되는 Change Journal Software의 버전은 2.0)
0x08	8	MFT Reference Number	현재 변경 이벤트가 적용되는 파일 혹은 디렉터리의 MFT Reference Number
0x10	8	Parent MFT Reference Number	현재 변경 이벤트가 적용되는 파일 혹은 디렉터리의 부모 디렉터리의 MFT Reference Number $MFT 정보와 조합하여 전체 경로 획득 가능
0x18	8	USN	Update Sequence Number
0x20	8	TimeStamp(FILETIME)	이벤트가 발생한 시간(UTC +0)
0x28	4	Reason Flag	변경 이벤트 정보 플래그
0x2C	4	Source Information	변경 이벤트를 발생시킨 주체에 대한 정보
0x30	4	Security ID	보안 ID
0x34	4	File Attributes	변경 이벤트의 대상이 되는 객체에 대한 정보 일반적으로 대상이 파일인지 디렉터리인지 구분
0x38	2	Size of Filename	객체 이름 정보의 크기
0x3A	2	Offset to Filename	객체 이름 정보의 레코드 내 위치
0x3C	N	Filename	현재 변경 이벤트가 적용되는 객체(파일 혹은 디렉터리)의 이름

- MFT Reference Number 대신 Parent MFT Reference Number 를 사용하는 이유
  - ✓ MFT Reference Number 를 사용할 경우, 해당 파일이 삭제되었을 때 전체 경로를 못 얻을 수도 있기 때문

$UsnJrnl 구조

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

Flag	Description
0x01	기본 $Data 속성에 데이터가 Overwrite 됨
0x02	기본 $Data 속성에 데이터가 추가됨
0x04	기본 $Data 속성에 데이터가 줄어듬
0x10	이름 있는 $Data 속성에 데이터가 Overwrite 됨
0x20	이름 있는 $Data 속성에 데이터가 추가됨
0x40	이름 있는 $Data 속성에 데이터가 줄어듬
0x100	파일이나 디렉터리가 생성됨
0x200	파일이나 디렉터리가 삭제됨
0x400	파일의 확장된 속성이 변경됨
0x800	접근 권한이 변경됨
0x1000	객체명 변경시, 변경 전 이름
0x2000	객체명 변경시, 변경 후 이름
0x4000	인덱스 상태가 변경됨
0x8000	파일이나 디렉터리의 속성이 변경됨
0x10000	하드 링크가 생성되었거나 삭제됨
0x20000	압축 상태가 변경됨(압축됨 or 압축이 풀림)
0x40000	암호화 상태가 변경됨(암호화됨 or 복호화됨)
0x80000	객체 ID가 변경됨
0x100000	Reparse 지점값이 변경됨
0x200000	이름 있는 $Data 속성의 생성 or 삭제 or 변경됨
0x80000000	파일 또는 디렉터리가 닫힘

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

Flag	Description
0x00	사용자가 발생시킨 이벤트
0x01	운영체제에 의해 발생한 이벤트
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.

$UsnJrnl 구조

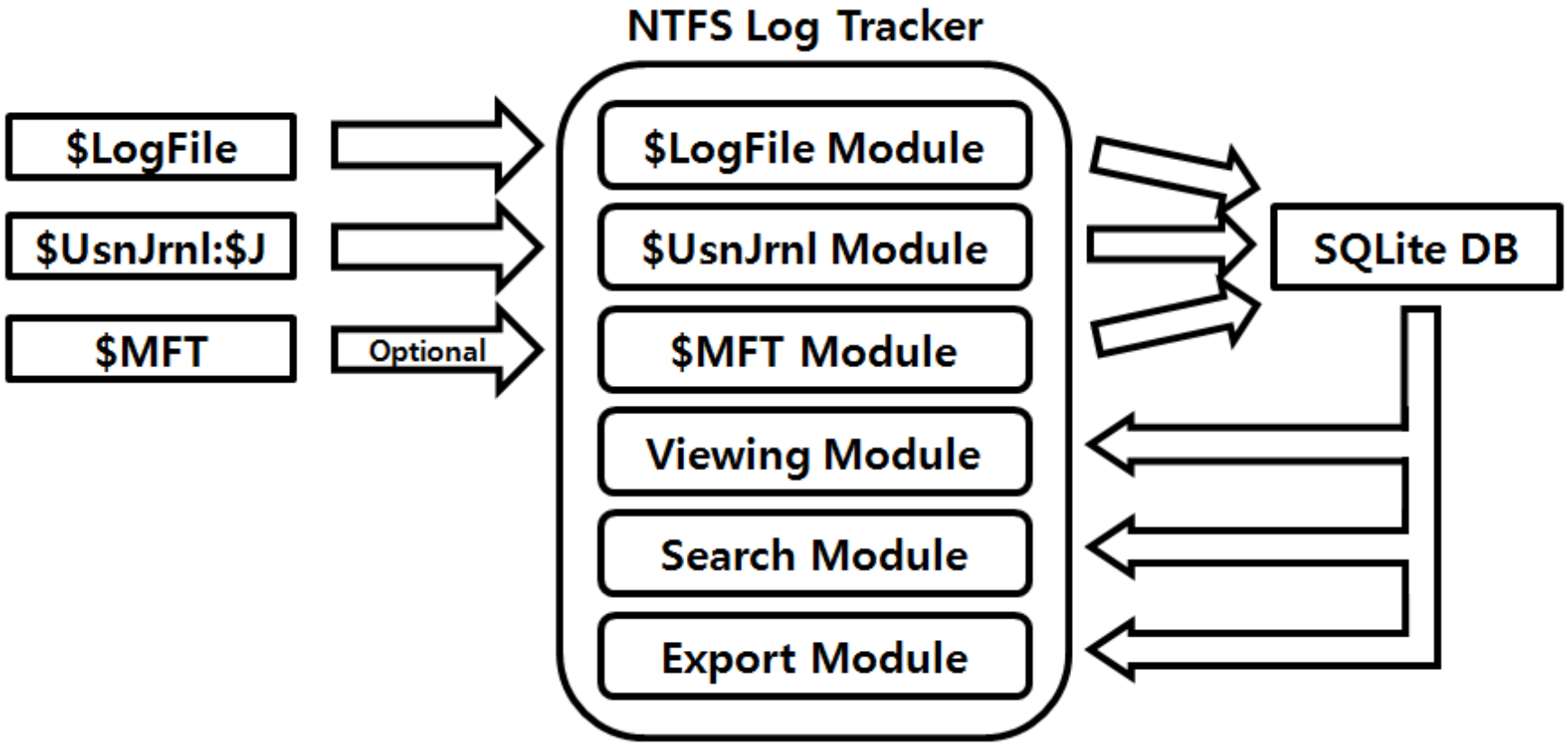
- **File Attribute 정보(**http://msdn.microsoft.com/en-us/library/gg258117.aspx**)**

Value	Description
0x01	읽기 전용 속성
0x02	숨김 속성
0x04	시스템 파일
0x10	디렉터리
0x20	Archive 파일
0x40	디바이스 파일
0x80	일반 파일
0x100	임시 파일
0x200	Sparse 파일
0x400	Reparse 속성을 가지고 있거나 심볼릭 링크 파일
0x800	압축됨
0x1000	This attribute indicates that the file data is physically moved to offline storage.
0x2000	인덱싱 안됨
0x4000	암호화됨
0x8000	The directory or user data stream is configured with integrity (only supported on ReFS volumes).
0x10000	가상 파일
0x20000	The user data stream not to be read by the background data integrity scanner (AKA scrubber).

NTFS Log Tracker

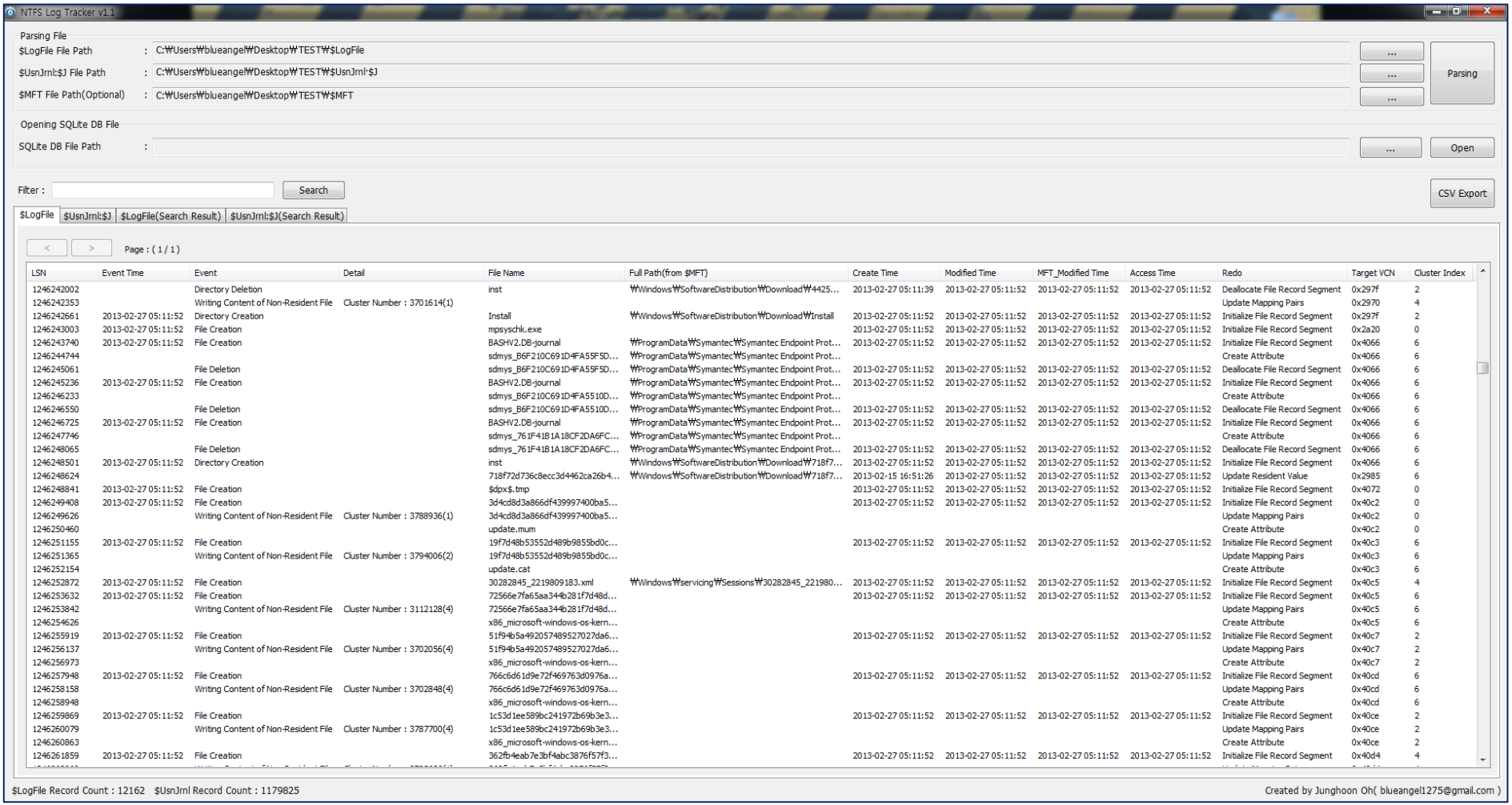
- **도구 설계 및 구현**
- 도구 기능
- 기존 도구와의 비교
- Case Study

도구 설계


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

NTFS Log Tracker

도구 구현 : https://code.google.com/p/ntfs-log-tracker/



NTFS Log Tracker

- 도구 설계
- **도구 기능**
- 기존 도구와의 비교
- Case Study

도구 기능

- **$LogFile 파일 단위 이벤트 추출**
  - **파일 생성/삭제 이벤트(파일시스템 터널링 포함)**

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

✓ 이벤트들 중 중간에 시간이 이상한 이벤트들은 "**파일 시스템 터널링**" 이벤트로 판단해주세요~^^

  - **파일 데이터 작성 이벤트**

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

  - **파일명 변경/이동 이벤트**

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

  - 추가적으로 $LogFile와 $MFT 에서 **LSN 이 겹치는 레코드들은 출력함**(파일명 포함)

도구 기능

- **$UsnJrnl 로그 출력**
  - 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

도구 기능

- 키워드 검색 기능
- CSV Export 기능
- SQLite DB Import 기능

NTFS Log Tracker

- 도구 설계
- 도구 기능
- **기존 도구와의 비교**
- Case Study

기존 도구와 비교

- **JP(Windows Journal Parser) : http://tzworks.net/prototype_page.php?proto_id=5**
  - Full Path 출력
    - ✓ JP 는 Full Path 정보를 출력해 주지 않음
  - 파일명 변경 이벤트

• 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

  - 파일/디렉터리 구분

• 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

기존 도구와 비교

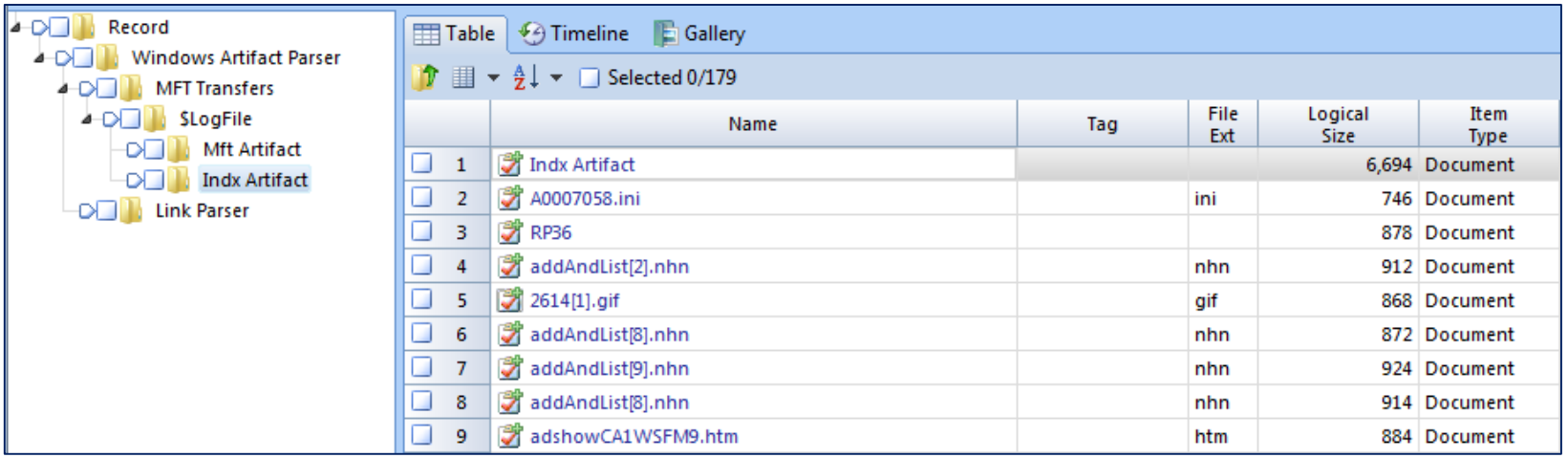
- **$LogFileParser :** https://code.google.com/p/mft2csv/wiki/LogFileParser
  - $LogFile, $UsnJrnl 레코드 단위 파싱
  - Data Run 추적
  - Full Path 정보 없음
  - 현장 분석용이기 보다는 연구용

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

기존 도구와 비교

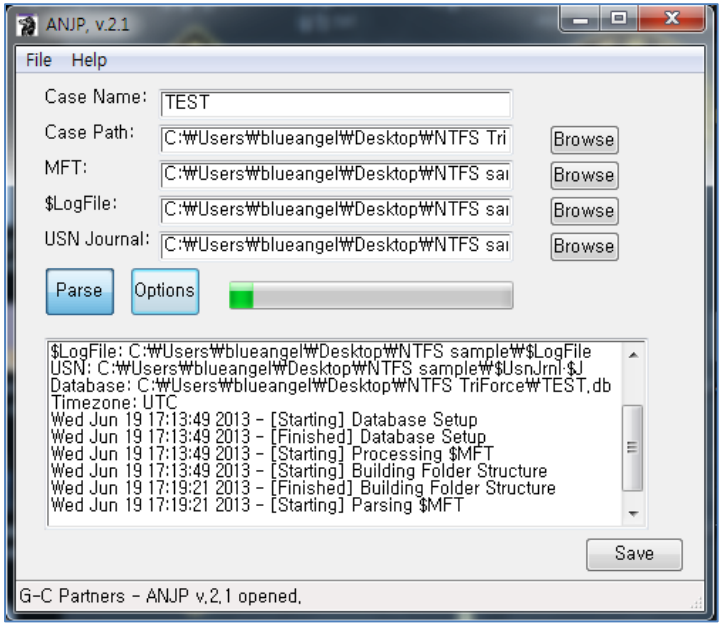
- **Encase v7**
  - MFT Transaction 분석 기능
    - ✓ $LogFile 내의 MFT Entry, Index Record 카빙
    - ✓ 파일 단위의 이벤트 정보를 추출하지 않음



	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

기존 도구와 비교

- **NTFS TriForce(** https://docs.google.com/forms/d/1GzOMe-QHtB12ZnI4ZTjLA06DJP6ZScXngO42ZDGIpR0/viewform)
  - $MFT, $LogFile, $UsnJrnl 교차 분석
  - 생성, 삭제, 이름 변경 이벤트 출력
  - SQLite, CSV 파일 출력



- **X-Ways Forensics**
  - $LogFile Viewer
  - 상용도구라 아직 써보질 못했음...

NTFS Log Tracker

- 도구 설계
- 도구 기능
- 기존 도구와의 비교
- **Case Study**

Case Study 1

- **부팅시 생성되고 지워지는 악성코드 추출**
  - 부팅시 생성되는 드라이버 파일 발견
  - 해당 파일은 $MFT 에서 흔적을 찾을 수 없음(로딩되고 지워져 있는 상태)
  - Cluster Number 정보를 통해 디스크 비할당영역에서 해당 드라이버 파일 추출
  - 리버싱을 통해 해당 드라이버의 정확한 역할을 알 수 있었음

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

- **메모리 내에서만 존재하는 악성코드 흔적 확인**
  - 악성코드가 메모리 내에서만 존재
  - 시스템 종료 이벤트를 탐지하여 Reloading 파일 생성
  - 부팅 후, 메모리에 로딩 된 후, Reloading 파일 삭제;;
  - 해당 파일은 $MFT 에서 흔적을 찾을 수 없음
  - $UsnJrnl 과 이벤트 로그를 교차 분석하여 종료될 때 생성되고 부팅될 때 삭제되는 파일 발견

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

- **Domain Controller(Win2008 R2)의 $UsnJrnl 분석**
  - 일반적 2008 R2 서버의 경우, 1~2일 정도의 변경 로그가 남음
  - DC(Domain Controller)의 경우, 1달 이상의 로그가 기록되어 있음(이유 모름;;)
  - DC의 $UsnJrnl 에서 공격자가 사용하는 악성코드 흔적을 찾기가 용이함
    - ✓ 획득한 키워드는 타 시스템 분석에 활용

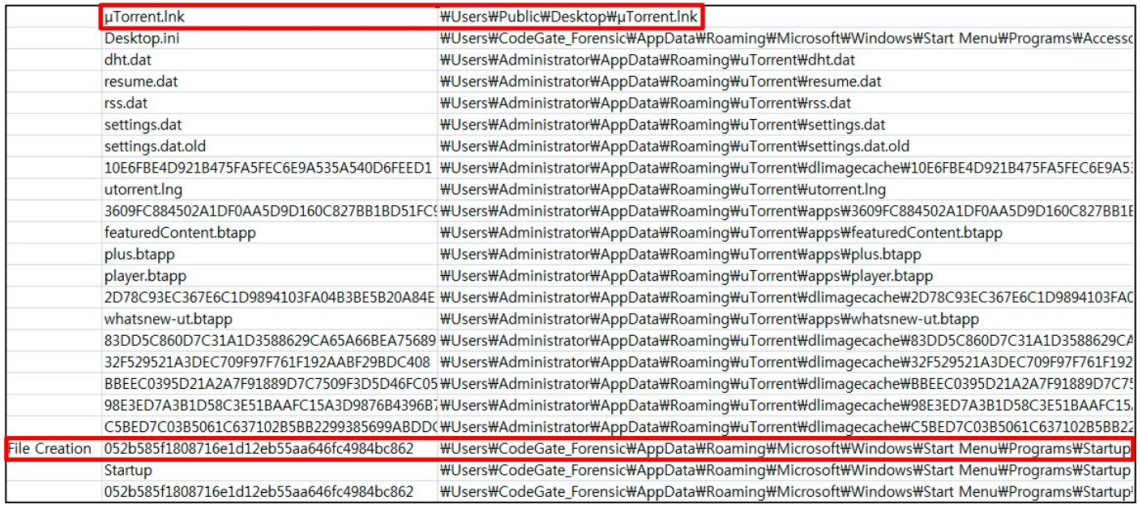
$LogFile | $UsnJrnl:$J | $LogFile(Search Result) | $UsnJrnl:$J(Search Result)

< > 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 문제 풀이 활용(** thanks to **Deok9~)**
  - 2013 CodeGate CTF, Forensic 200
  - 문제에서 주어진 이미지의 $LogFile 분석
    - ✓ 특정 경로에 생성되는 파일 발견



	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

    - ✓ 문제를 만드는 동안의 모든 파일 시스템 이벤트를 확인 가능

  - 자세한 문제 풀이는 아래 URL 에서 확인

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

Conclusion

Conclusion

- **NTFS 의 로그 파일** : $LogFile, $UsnJrnl
- **$MFT 에만 의존한 파일 시스템 분석은 한계가 있음**
  - 삭제된 파일의 흔적
  - 특정 파일의 동일한 이벤트
- **$LogFile, $UsnJrnl 을 통한 파일 시스템 이벤트 분석이 필요함**
- **NTFS Log Tracker**
  - $LogFile, $UsnJrnl 이벤트 분석
  - $MFT 를 통해 Full Path 추출
  - 키워드 검색, CSV Export, SQLite 지원

Question and Answer