A Dig into the $LogFile

blueangel

blueangel1275@gmail.com

http://blueangel-forensic-note.tistory.com

목차

서론

- **$LogFile 이란?**
- $LogFile 크기 조절

$LogFile 이란?

- **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

서론

- $LogFile 이란?
- **$LogFile 크기 조절**

$LogFile 크기 조절

- **$LogFile 크기**
  - 일반적인 하드디스크 볼륨에서는 64M 크기
  - 볼륨 용량에 따라 크기가 달라질 수 있지만 기본적으로는 최대 64M 이하임

- **크기 조절**
  - chkdsk 명령의 /L 옵션에 따라 크기 조절 가능
  - "/L : 파일크기(KB 단위)" 형식으로 지정
  - 크기가 지정 되지 않으면 현재 크기 표시

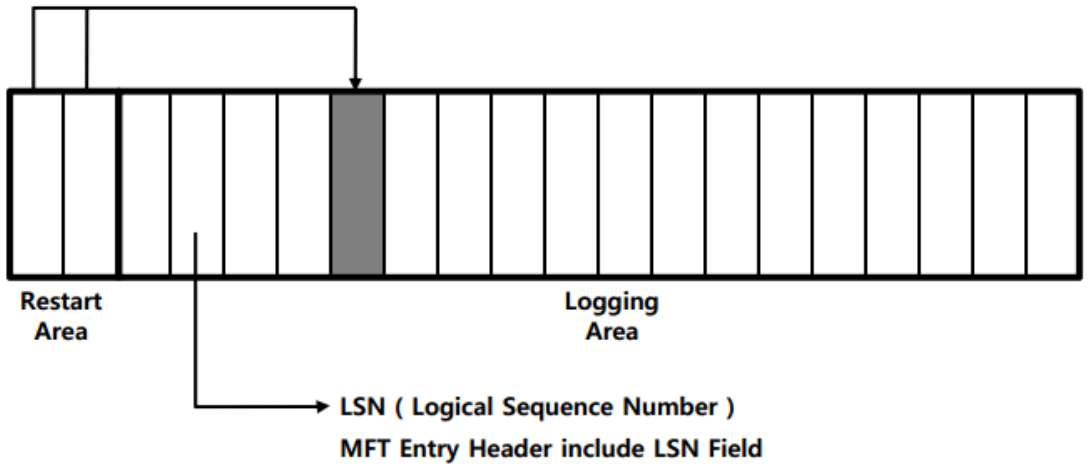
```
C:₩>chkdsk /L
파일 시스템 유형은 NTFS입니다.
현재 로그 파일 크기는 65536KB입니다.
이 볼륨의 기본 로그 파일 크기는 65536KB입니다.
```

$LogFile 구조

- **전체 구조**
- 재시작 영역 구조
- 로깅 영역 구조
- 페이지 구조
- 레코드 구조

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



$LogFile 구조

- 전체 구조
- **재시작 영역 구조**
- 로깅 영역 구조
- 페이지 구조
- 레코드 구조

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**

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			

$LogFile 구조

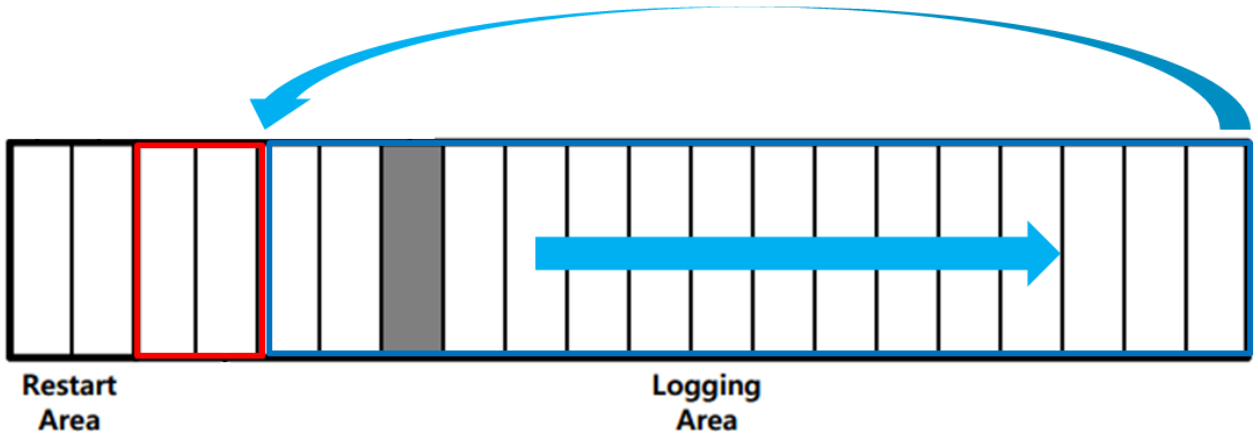
- 전체 구조
- 재시작 영역 구조
- **로깅 영역 구조**
- 페이지 구조
- 레코드 구조

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~)
    ➔ 순차적으로 레코드가 기록됨
    ➔ 파일 끝까지 기록되면 다시 영역 앞에서부터 덮어씀



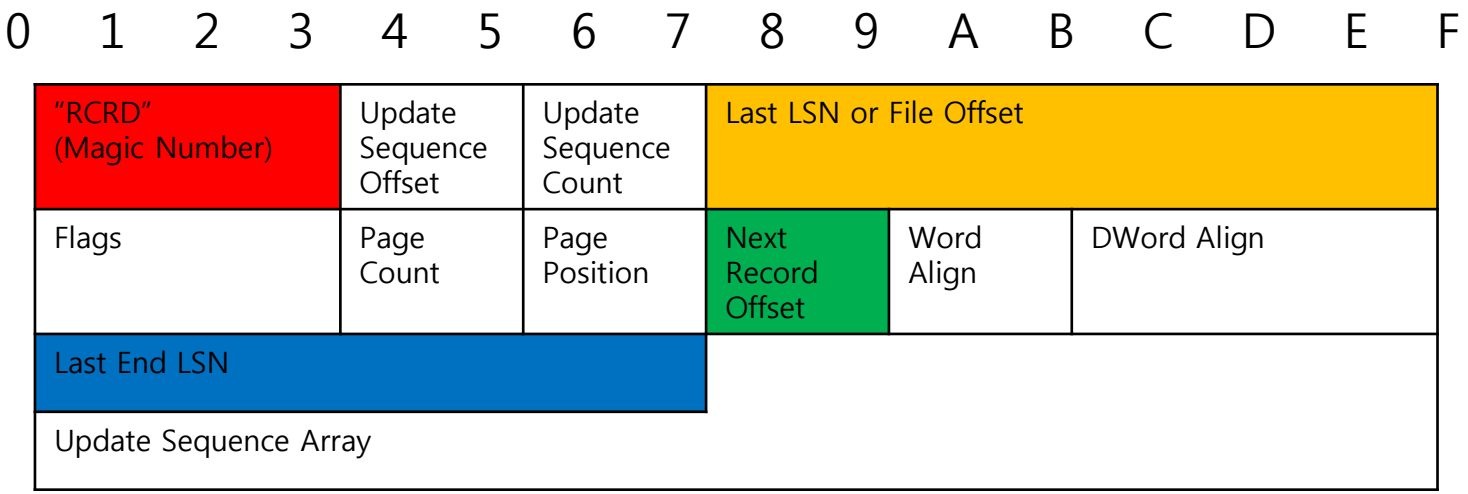
$LogFile 구조

- 전체 구조
- 재시작 영역 구조
- 로깅 영역 구조
- **페이지 구조**
- 작업 레코드 구조

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

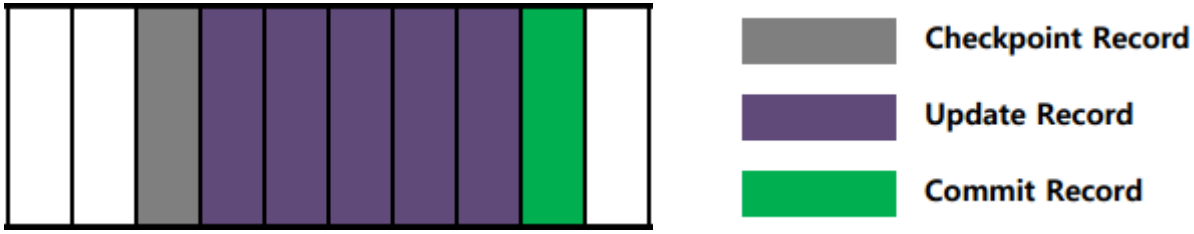


$LogFile 구조

- 전체 구조
- 재시작 영역 구조
- 로깅 영역 구조
- 페이지 구조
- **작업 레코드 구조**

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

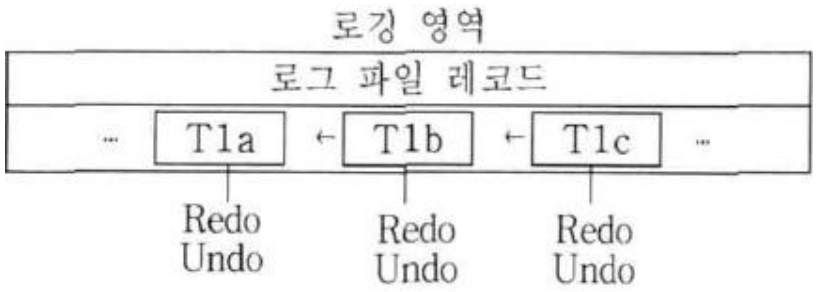


  - Check Point Record 외 레코드들은 자신의 이전 작업 레코드의 LSN을 가지고 있음

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
- Taret LCN : Redo/Undo 데이터가 적용되는 디스크 상의 LCN(Logical Cluster Number)
- 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

- **Redo/Undo 연산 코드**

NTFS 로그 동작	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 이벤트 분석

- **파일 생성 이벤트**
- 파일 삭제 이벤트
- 파일 데이터 작성/수정 이벤트
- 파일명 변경 이벤트

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

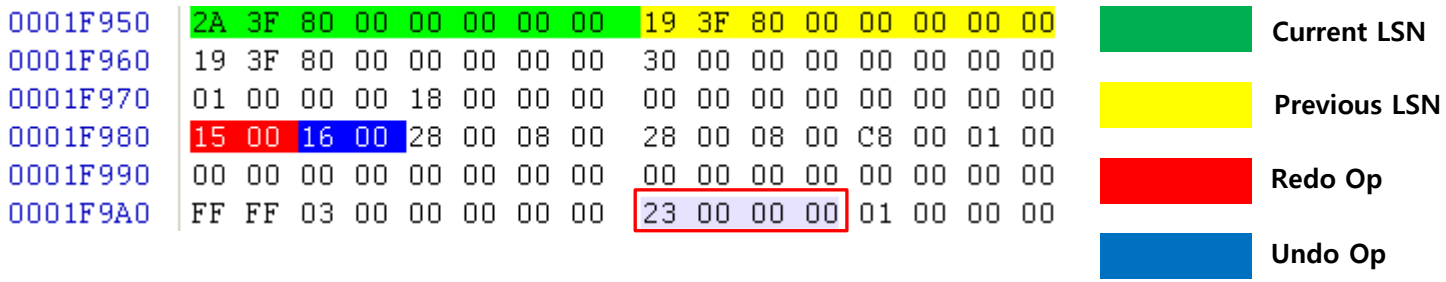
LSN	Previous LSN	Record Type	Event	Detail	File Name	Full Path	Redo	Undo
8405882	0	Check Point Record					Noop	Noop
8405901	0	Update Record					Set Bits In Nonresident Bit Map	Clear Bits In Nonresident Bit Map
8405913	8405901	Update Record					Noop	Deallocate File Record Segment
8405924	8405913	Update 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 File 생성 이벤트 순서(Redo/Undo) : 중간에 들어갈 수 있는 OpenNonResidentAttribute Redo 작업은 무시
  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)

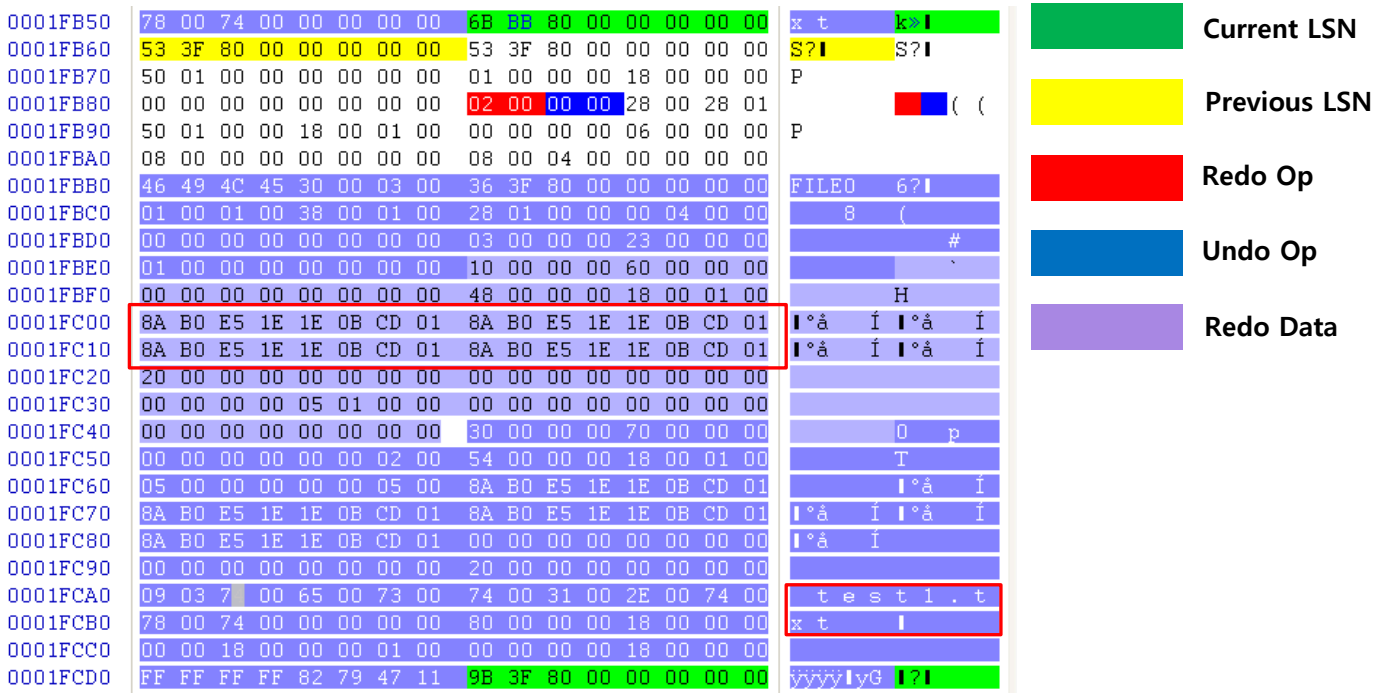
파일 생성 이벤트

- **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 시간과 생성 파일명, 전체 경로
    - ✓ 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 경로를 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

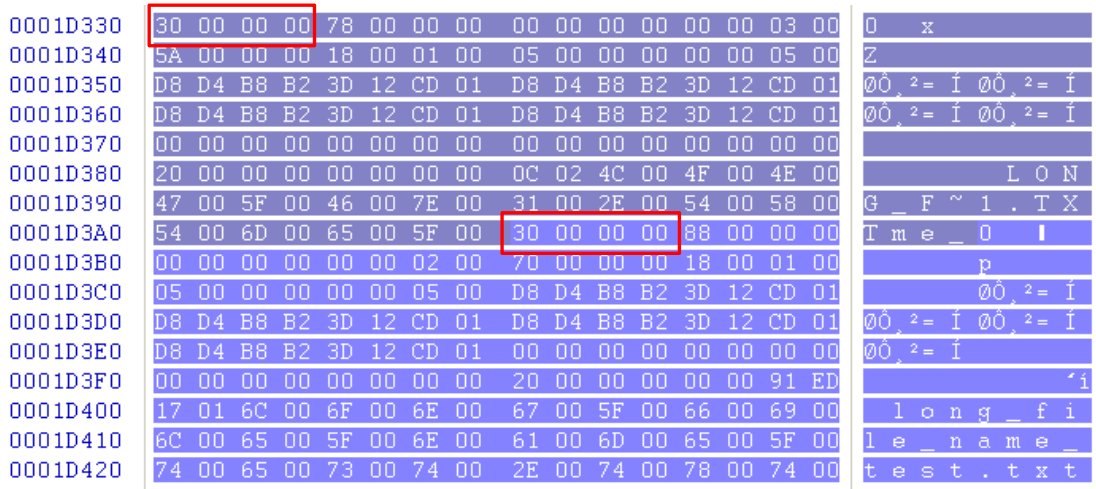
LSN	Previous LSN	Record Type	Event	Detail	Redo	Undo
8407193	8407176	Update Record			Set Bits In Nonresident Bit Map	Clear Bits In Nonresident Bit Map
8407205	8407193	Update Record			Noop	Deallocate File Record Segment
8407216	8407205	Update Record			Add Index Entry Allocation	Delete Index Entry Allocation
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



$LogFile 이벤트 분석

- 파일 생성 이벤트
- **파일 삭제 이벤트**
- 파일 데이터 작성/수정 이벤트
- 파일명 변경 이벤트

파일 삭제 이벤트

- **Resident File 삭제 관련 이벤트**

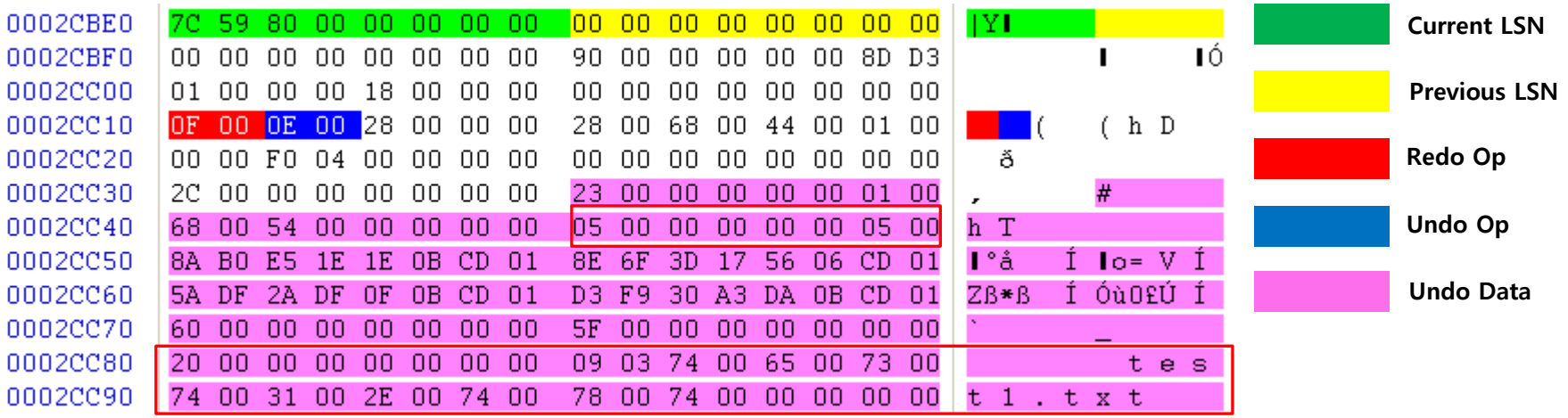
8411497	0	Check Point Record	Noop	Noop
8411516	0	Update Record	Delete Index Entry Allocation	Add Index Entry Allocation
8411540	8411516	Update Record	Deallocate File Record Segment	Initialize File Record Segment
8411555	8411540	Update Record	OpenNonresidentAttribute	Noop
8411572	8411555	Update Record	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) : 중간에 들어갈 수 있는 OpenNonResidentAttribute Redo 작업은 무시
  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 파일명, 전체 경로**
    - ✓ 0x0F/0x0E(Delete Index Entry Allocation/Add Index Entry Allocation) 작업의 Undo 데이터에서 얻어옴
    - ✓ Undo 데이터의 내용은 Index Entry 안의 Content 내용($FileName 속성)
    - ✓ Parent File Reference Address 값을 통해 부모 디렉토리와 전체 경로를 가져옴
    - ✓ Name 값을 통해 삭제된 파일명 획득

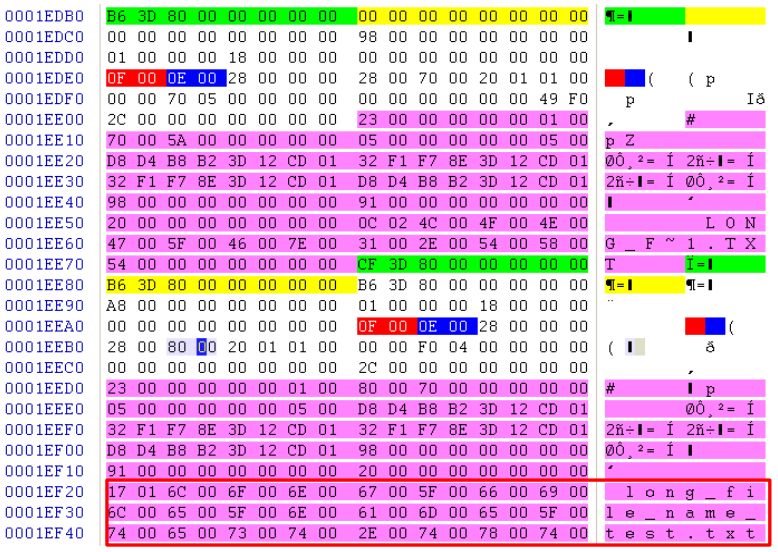


파일 삭제 이벤트

- **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) : "Delete Index Entry Root" Redo 작업이 들어갈 수 있음
  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)

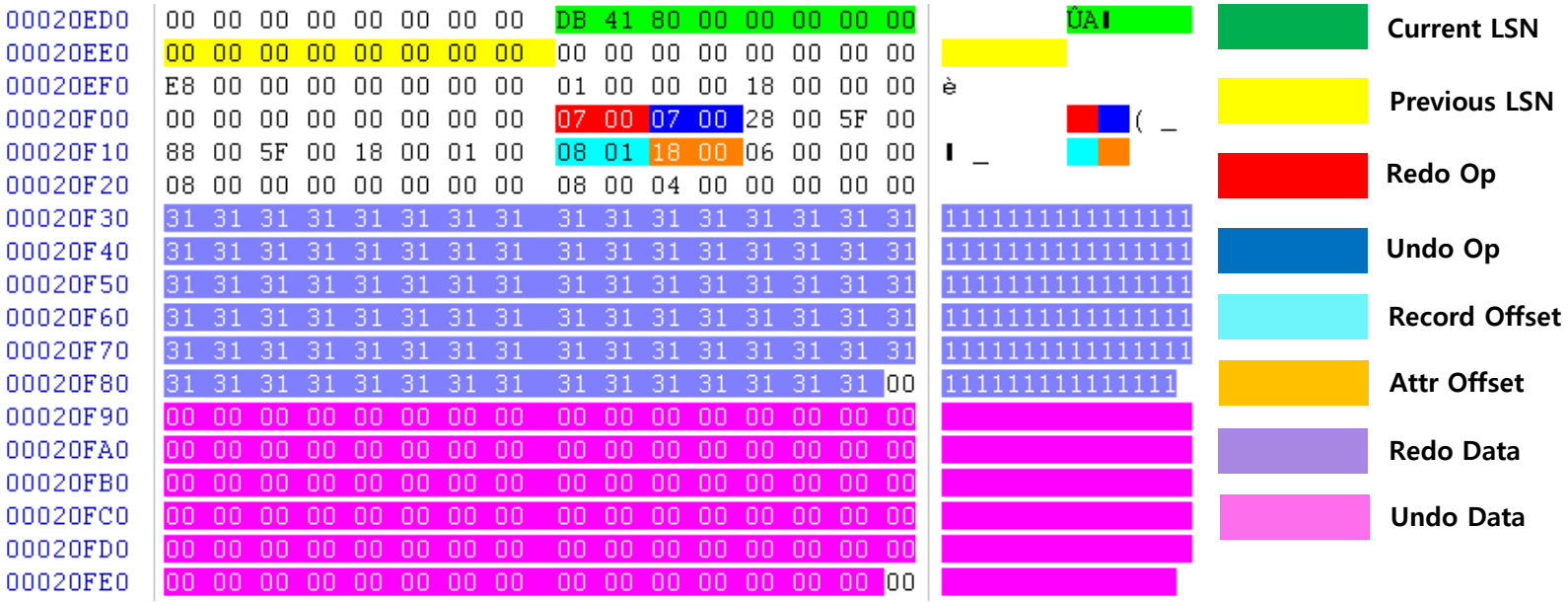
$LogFile 이벤트 분석

- 파일 생성 이벤트
- 파일 삭제 이벤트
- **파일 데이터 작성/수정 이벤트**
- 파일명 변경 이벤트

파일 데이터 작성/수정 이벤트

- **Resident File 파일 데이터 작성**
  - 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/수정 이벤트

- **Resident File 파일 데이터 수정**
  - 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

```
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

파일 데이터 작성/수정 이벤트

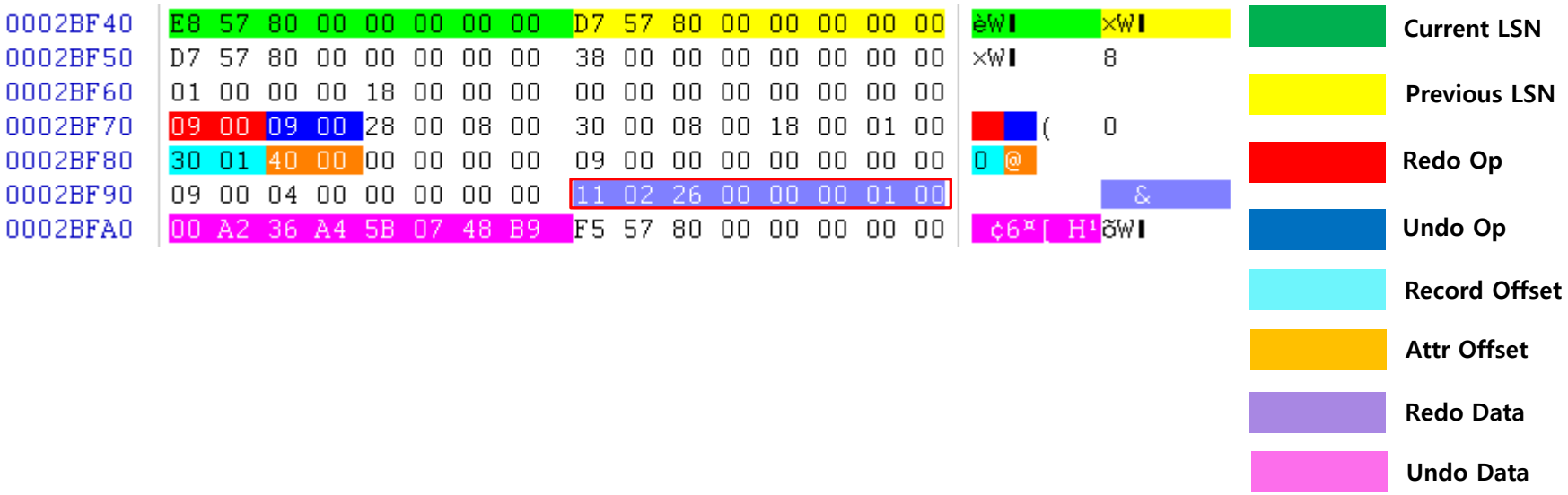
- **대상 파일 찾기**
  - Update Resident Value 작업의 Target LCN, MFT Cluster Index 값과 Initialize File Record Segment 작업의 Target LCN, MFT Cluster Index 값을 비교
  - 같은 Target LCN, MFT Cluster Index 값을 가지고 있으면 Initialize File Record Segment 작업을 통해 생성된 파일의 내용을 작성/수정한 것이라 볼 수 있음

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/수정 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 확인 할 수 있음
    - ✓ Attr Offset이 0x40 일 경우, Cluster Run 작성 내용을 Redo/Undo 데이터에서 획득할 수 있음 (0x41일 경우, 확인 불가)
    - ✓ Redo/Undo 작업의 데이터는 Cluster Run 작성 내용임
      - ➔ 아래의 경우, 0x26번째 클러스터부터 2클러스터가 사용되었음



파일 데이터 작성/수정 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)

$LogFile 이벤트 분석

- 파일 생성 이벤트
- 파일 삭제 이벤트
- 파일 데이터 작성/수정 이벤트
- **파일명 변경 이벤트**

파일명 변경 이벤트

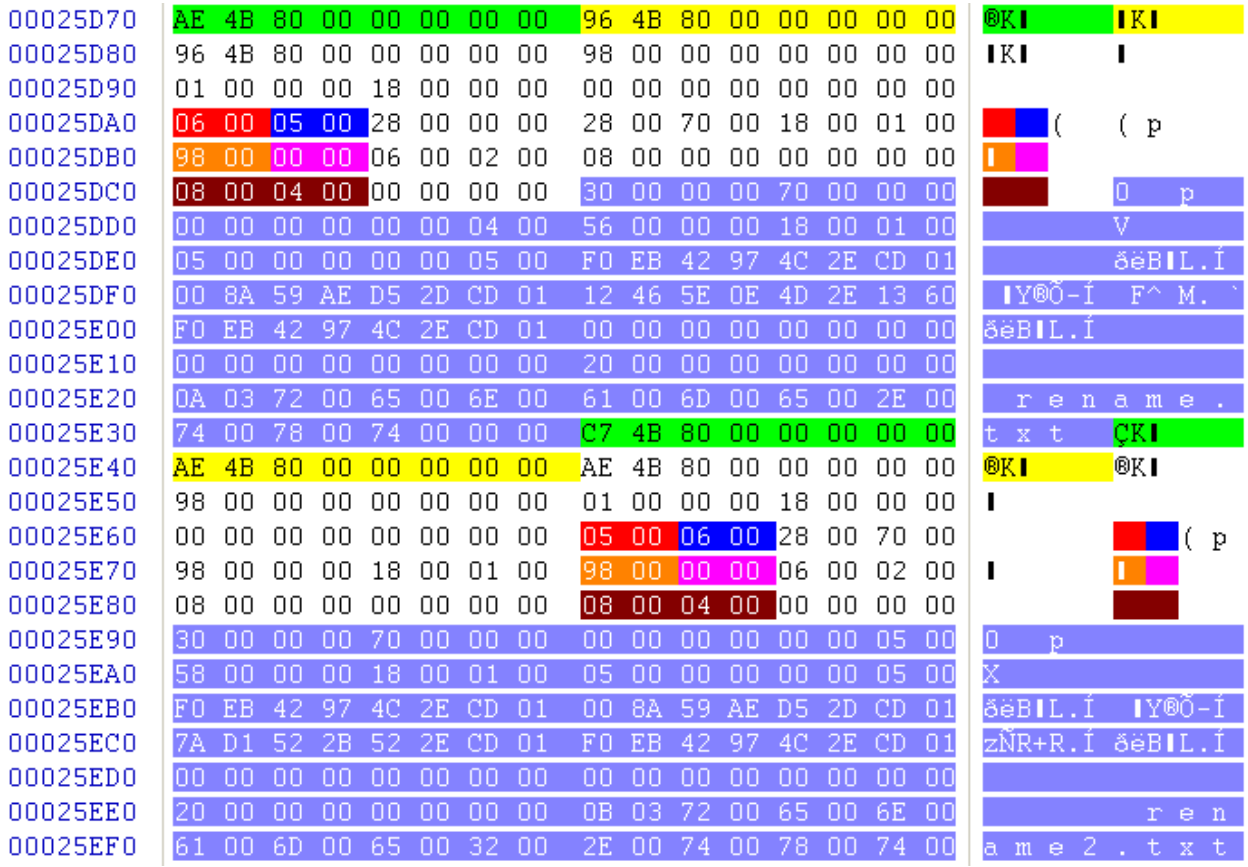
- **파일명 변경 시, 일어나는 작업**
  - 인덱스 삭제, 추가 작업
  - $FILE_NAME 속성 삭제, 추가 작업
    - ✓ Record Offset 이 0x98, Attr Offset 이 0x00 인 Delete Attribute와 Create Attribute 작업이 연속적으로 오면 파일명 변경 ➔ 일반적으로 $FILE_NAME 속성은 MFT 레코드에서 0x98 위치에 있음
    - ✓ 두 작업의 Target LCN이 동일해야 함

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 에서 변경 전 파일명과 변경 후 파일명을 알 수 있음



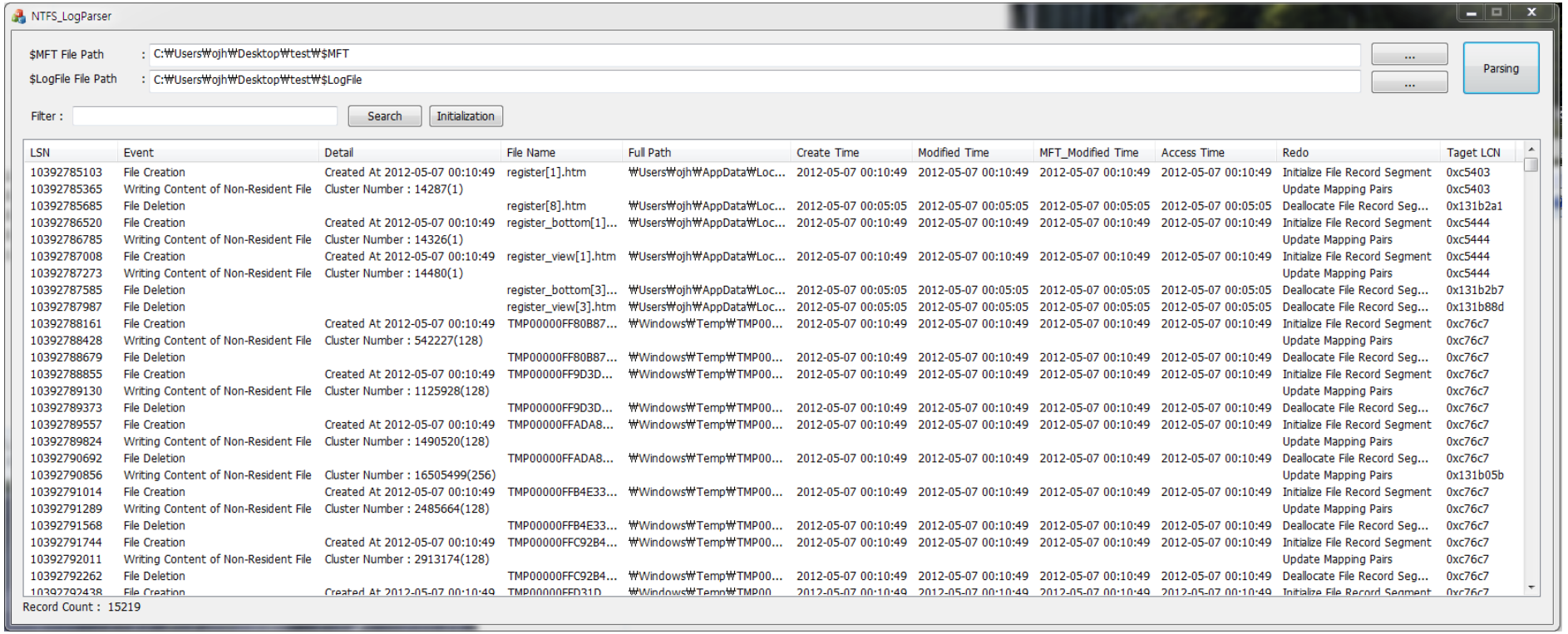
$LogFile Parser 구현

- **도구 설계**
- 기능 설명

$LogFile Parser 구현

도구 설계

1. $MFT, $LogFile 을 입력으로 받음
2. $LogFile 작업 레코드의 LSN과 $MFT의 레코드의 LSN 비교하여 파일명/전체 경로/시간정보 획득
3. $LogFile 작업 레코드 파싱 중에 얻는 Parent File Reference Address 값은 $MFT 모듈에 넘겨 전체 경로 획득
4. $LogFile 작업 레코드를 파싱하여 파일 생성, 삭제, 데이터 작성, 파일명 변경 이벤트 획득



$LogFile Parser 구현

- 도구 설계
- **기능 설명**

기능 설명

- **파일 생성 이벤트 출력**
  - Redo 작업 데이터에서 파일명, 생성 시간 획득
  - Redo 작업 데이터의 Parent File Reference Address 값은 $MFT 모듈에 넘겨 전체 경로 획득

LSN	Event	Detail	File Name	Full Path	Create Time	Modified Time	MFT_Modified Time	Access Time	Redo	Taget LCN
10409540182	File Creation	Created At 2012-01-10 23:59:15	MSStmp.log	₩ProgramData₩Microsoft₩...	2012-01-10 23:59:15	2012-05-08 01:32:00	2012-05-08 01:32:00	2012-05-08 01:32:00	Initialize File Record Segment	0xc48fd
10409540929	File Creation	Created At 2012-05-08 01:32:02	TMP0000001BE10...	₩Windows₩Temp₩TMP00...	2012-05-08 01:32:02	2012-05-08 01:32:02	2012-05-08 01:32:02	2012-05-08 01:32:02	Initialize File Record Segment	0xc4a1f
10409541623	File Creation	Created At 2012-05-08 01:32:02	TMP0000001CF28...	₩Windows₩Temp₩TMP00...	2012-05-08 01:32:02	2012-05-08 01:32:02	2012-05-08 01:32:02	2012-05-08 01:32:02	Initialize File Record Segment	0xc4a1f
10409542158	File Creation	Created At 2012-05-08 01:32:02	01cd2c6eee7b299b	₩Windows₩Temp₩01cd2c...	2012-05-08 01:32:02	2012-05-08 01:32:02	2012-05-08 01:32:02	2012-05-08 01:32:02	Initialize File Record Segment	0xc4b79
10409543120	File Creation	Created At 2012-05-08 01:32:04	NTFS_LOGPARSER....	₩Windows₩Prefetch₩NTF...	2012-05-08 01:32:04	2012-05-08 01:32:04	2012-05-08 01:32:04	2012-05-08 01:32:04	Initialize File Record Segment	0xc4b79
10409543743	File Creation	Created At 2012-05-08 01:32:05	{40BA8EA2-B87C-...	₩ProgramData₩Microsoft₩...	2012-05-08 01:32:05	2012-05-08 01:32:05	2012-05-08 01:32:05	2012-05-08 01:32:05	Initialize File Record Segment	0xc5317
10409547841	Directory Creation	Created At 2012-05-08 01:32:22	해오라기	₩Users₩ojh₩Desktop₩해...	2012-05-08 01:32:22	2012-05-08 01:32:22	2012-05-08 01:32:22	2012-05-08 01:32:22	Initialize File Record Segment	0xc537d
10409548808	File Creation	Created At 2012-05-08 01:32:28	$MFT	₩Users₩ojh₩Desktop₩tes...	2012-05-08 01:32:28	2012-05-08 01:32:28	2012-05-08 01:32:28	2012-05-08 01:32:28	Initialize File Record Segment	0xc4a1f
10409559028	File Creation	Created At 2012-05-08 01:32:30	$LogFile	₩Users₩ojh₩Desktop₩tes...	2012-05-08 01:32:30	2012-05-08 01:32:30	2012-05-08 01:32:30	2012-05-08 01:32:30	Initialize File Record Segment	0xc5383

- **파일 삭제 이벤트 출력**
  - Undo 작업 데이터에서 파일명 획득
  - Undo 작업 데이터의 Parent File Reference Address 값은 $MFT 모듈에 넘겨 전체 경로 획득
  - 삭제 시간은 알 수 없음 (앞 뒤 이벤트 발생 시간을 통해 대략적인 삭제 시간 유추)

LSN	Event	Detail	File Name	Full Path	Create Time	Modified Time	MFT_Modified Time	Access Time	Redo	Taget LCN
10409402178	File Deletion		nmnt_count[1].htm	₩Users₩ojh₩AppData₩Loc...	2012-05-08 01:31:08	2012-05-08 01:31:08	2012-05-08 01:31:08	2012-05-08 01:31:08	Deallocate File Record Seg...	0xc7057
10409417743	File Deletion		TMP00000019037...	₩Windows₩Temp₩TMP00...	2012-05-08 01:31:39	2012-05-08 01:31:39	2012-05-08 01:31:39	2012-05-08 01:31:39	Deallocate File Record Seg...	0xc70dd
10409465710	File Deletion		TMP0000001A9F8...	₩Windows₩Temp₩TMP00...	2012-05-08 01:31:45	2012-05-08 01:31:45	2012-05-08 01:31:45	2012-05-08 01:31:45	Deallocate File Record Seg...	0xc70df
10409535635	File Deletion		bplist.dll	₩Users₩ojh₩Desktop₩Rel...	2012-02-01 23:22:50	2011-06-15 15:26:44	2012-02-01 23:22:50	2012-02-01 23:22:50	Deallocate File Record Seg...	0xc537d
10409535793	File Deletion		carvIDX_2.exe	₩Users₩ojh₩Desktop₩Rel...	2012-02-01 23:22:50	2011-06-07 11:17:42	2012-02-01 23:22:50	2012-02-01 23:22:50	Deallocate File Record Seg...	0xc5382
10409535952	File Deletion		chartdir50.dll	₩Users₩ojh₩Desktop₩Rel...	2012-02-01 23:22:50	2009-03-28 23:59:36	2012-02-01 23:22:50	2012-02-01 23:22:50	Deallocate File Record Seg...	0xc5383
10409536110	File Deletion		Language.conf	₩Users₩ojh₩Desktop₩Rel...	2012-02-01 23:22:50	2012-02-12 23:43:11	2012-02-12 23:43:11	2012-02-01 23:22:50	Deallocate File Record Seg...	0xc5384

기능 설명

- **파일 데이터 작성 이벤트 출력**
  - Resident 파일 데이터 작성 이벤트
    - ✓ 작성된 파일 데이터의 $LogFile 파일 내 Offset 값을 출력(추후에 헥사값 출력으로 업데이트)
    - ✓ 일반적으로 생성 이벤트 바로 뒤에 데이터 작성 이벤트가 따라서 나옴

Event	Detail	File Name	Full Path	Create Time	Modified Time	MFT_Modified Time	Access Time
File Creation	Created At 2012-03-26 16:00:25	test1.txt	₩test1.txt	2012-03-26 16:00:25	2012-03-26 16:00:25	2012-03-26 16:00:25	2012-03-26 16:00:25
Writing Content of Resident File	Data Offset : 134960	test1.txt	₩test1.txt	2012-03-26 16:00:25	2012-03-20 13:58:29	2012-03-26 14:18:26	2012-03-26 16:00:25
File Creation	Created At 2012-03-26 16:02:07	test2.txt	₩test2.txt	2012-03-26 16:02:07	2012-03-26 16:02:07	2012-03-26 16:02:07	2012-03-26 16:02:07
Writing Content of Resident File	Data Offset : 141528	test2.txt	₩test2.txt	2012-03-26 16:02:07	2012-03-20 13:58:37	2012-03-26 14:18:26	2012-03-26 16:02:07
File Creation	Created At 2012-03-26 16:04:21	test3.txt	₩test3.txt	2012-03-26 16:04:21	2012-03-26 16:04:21	2012-03-26 16:04:21	2012-03-26 16:04:21
Writing Content of Resident File	Data Offset : 150376	test3.txt	₩test3.txt	2012-03-26 16:04:21	2012-03-20 13:58:45	2012-03-26 14:18:26	2012-03-26 16:04:21
File Creation	Created At 2012-03-26 16:08:31	test4.txt	₩test4.txt	2012-03-26 16:08:31	2012-03-26 16:08:31	2012-03-26 16:08:31	2012-03-26 16:08:31
Writing Content of Resident File	Data Offset : 157000	test4.txt	₩test4.txt	2012-03-26 16:08:31	2012-03-26 16:07:17	2012-03-26 16:07:17	2012-03-26 16:08:31
File Creation	Created At 2012-03-26 16:10:10	test5.txt	₩test5.txt	2012-03-26 16:10:10	2012-03-26 16:10:10	2012-03-26 16:10:10	2012-03-26 16:10:10

  - Non-Resident 파일 데이터 작성 이벤트
    - ✓ 파일 데이터 작성된 LCN 값 출력
    - ✓ 일반적으로는 생성 이벤트 바로 뒤에 데이터 작성 이벤트가 따라서 나옴

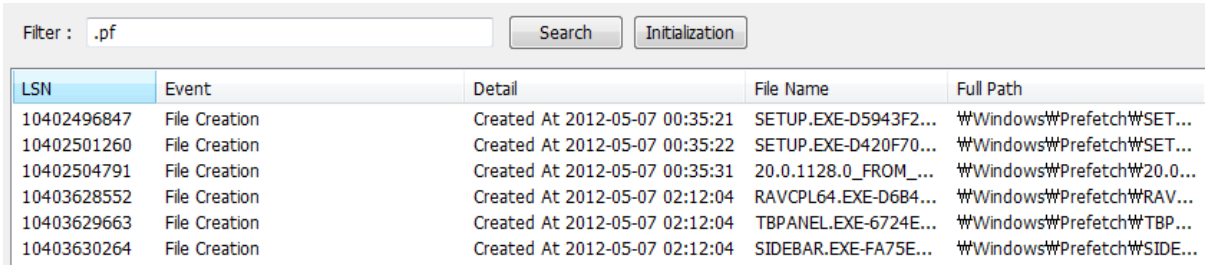
Event	Detail	File Name	Full Path	Create Time	Modified Time	MFT_Modified Time	Access Time
File Creation	Created At 2012-04-19 01:27:11	register_bottom[1].htm	₩Users₩ojh₩AppData₩Loc...	2012-04-19 01:27:11	2012-04-19 01:27:11	2012-04-19 01:27:11	2012-04-19 01:27:11
Writing Content of Non-Resident File	Cluster Number : 16671445(1)	register_bottom[1].htm	₩Users₩ojh₩AppData₩Loc...				
File Creation	Created At 2012-04-19 01:27:11	register_view[1].htm	₩Users₩ojh₩AppData₩Loc...	2012-04-19 01:27:11	2012-04-19 01:27:11	2012-04-19 01:27:11	2012-04-19 01:27:11
Writing Content of Non-Resident File	Cluster Number : 16675871(1)	register_view[1].htm	₩Users₩ojh₩AppData₩Loc...				
File Creation	Created At 2012-04-19 01:27:11	1334682183046PNhGXGCrHp[1].jpg	₩Users₩ojh₩AppData₩Loc...	2012-04-19 01:27:11	2012-04-19 01:27:11	2012-04-19 01:27:11	2012-04-19 01:27:11
Writing Content of Non-Resident File	Cluster Number : 2578263(50)	1334682183046PNhGXGCrHp[1].jpg	₩Users₩ojh₩AppData₩Loc...				
File Creation	Created At 2012-04-19 01:27:11	1334682300644PNhGXGCrHp[1].jpg	₩Users₩ojh₩AppData₩Loc...	2012-04-19 01:27:11	2012-04-19 01:27:11	2012-04-19 01:27:11	2012-04-19 01:27:11
Writing Content of Non-Resident File	Cluster Number : 1964233(16)	1334682300644PNhGXGCrHp[1].jpg	₩Users₩ojh₩AppData₩Loc...				

기능 설명

- **파일명 변경 이벤트 출력**
  - 변경 전 파일명과 변경 후 파일명 출력
  - MFT Modified Time을 통해 파일명 변경 시간 획득

LSN	Event	Detail	File Name	Full Path	Create Time	Modified Time	MFT_Modified Time	Access Time
10403274179	File/Directory Name Modification	Modified At 2012-05-07 02:11:22, Resource.new -> Resource.dat	Resource.dat	₩ProgramData₩NVIDIA₩R...	2012-05-07 02:11:22	2012-05-07 02:11:22	2012-05-07 02:11:22	2012-05-07 02:11:22
10403290900	File/Directory Name Modification	Modified At 2012-05-06 14:17:27, ERRORLOG.5 -> ERRORLOG.6	ERRORLOG.6	₩Program Files (x86)₩Micro...	2012-01-12 01:30:22	2012-05-02 01:33:20	2012-05-06 14:17:27	2012-05-01 23:41:36
10403291348	File/Directory Name Modification	Modified At 2012-05-06 14:17:27, ERRORLOG.4 -> ERRORLOG.5	ERRORLOG.5	₩Program Files (x86)₩Micro...	2012-01-12 01:30:22	2012-05-03 01:51:54	2012-05-06 14:17:27	2012-05-02 23:49:17
10403291752	File/Directory Name Modification	Modified At 2012-05-06 14:17:27, ERRORLOG.3 -> ERRORLOG.4	ERRORLOG.4	₩Program Files (x86)₩Micro...	2012-01-12 01:30:22	2012-05-04 03:53:50	2012-05-06 14:17:27	2012-05-04 02:51:48
10403292148	File/Directory Name Modification	Modified At 2012-05-06 14:17:27, ERRORLOG.2 -> ERRORLOG.3	ERRORLOG.3	₩Program Files (x86)₩Micro...	2012-01-12 01:30:22	2012-05-05 13:35:28	2012-05-06 14:17:27	2012-05-04 23:38:14
10403292552	File/Directory Name Modification	Modified At 2012-05-06 14:17:27, ERRORLOG.1 -> ERRORLOG.2	ERRORLOG.2	₩Program Files (x86)₩Micro...	2012-01-12 01:30:22	2012-05-06 05:38:25	2012-05-06 14:17:27	2012-05-05 22:41:27
10403292956	File/Directory Name Modification	Modified At 2012-05-07 02:08:56, ERRORLOG -> ERRORLOG.1	ERRORLOG.1	₩Program Files (x86)₩Micro...	2012-01-12 01:30:22	2012-05-07 02:08:56	2012-05-07 02:08:56	2012-05-06 14:17:27

- **키워드 검색**
  - 원하는 키워드가 포함된 이벤트만 필터링
  - 필터링 예
    - ✓ ".exe" or ".dll" or ".sys" ➔ PE 파일 행위 분석에 활용
    - ✓ ".lnk" ➔ 문서 열람 흔적 추적에 활용
    - ✓ ".pf" ➔ PE 파일 실행 흔적 추적에 활용



결론

$LogFile의 포렌식적 의미

- **NTFS 작업 히스토리 추적**
  - 파일/디렉토리 생성, 삭제, 이름 변경, MFT 수정 작업의 타임라인 작성

- **MFT 에서 발견하지 못한 삭제된 파일 흔적 추적**
  - 삭제된 파일의 MFT가 덮여졌을 경우라도 $LogFile 에는 삭제 기록이 남아 있음
  - Resident 파일의 경우, 파일 데이터 확인 가능
  - Non-Resident 파일의 경우, 파일 데이터가 작성된 클러스터 위치 확인 가능

- **문서 파일 열람 흔적, PE 파일 실행 및 생성/삭제 흔적 추적에 활용**

질문 및 답변