SHINING A LIGHT ON SPOTLIGHT

LEVERAGING APPLE'S DESKTOP SEARCH UTILITY TO RECOVER DELETED FILE METADATA ON MACOS

Taj Atwal

REDUCTECH

Mark Scanlon
Nhien-An Le-Khac

UCD DUBLIN

UCD Forensics and
Security Research Group

MAIN POINTS

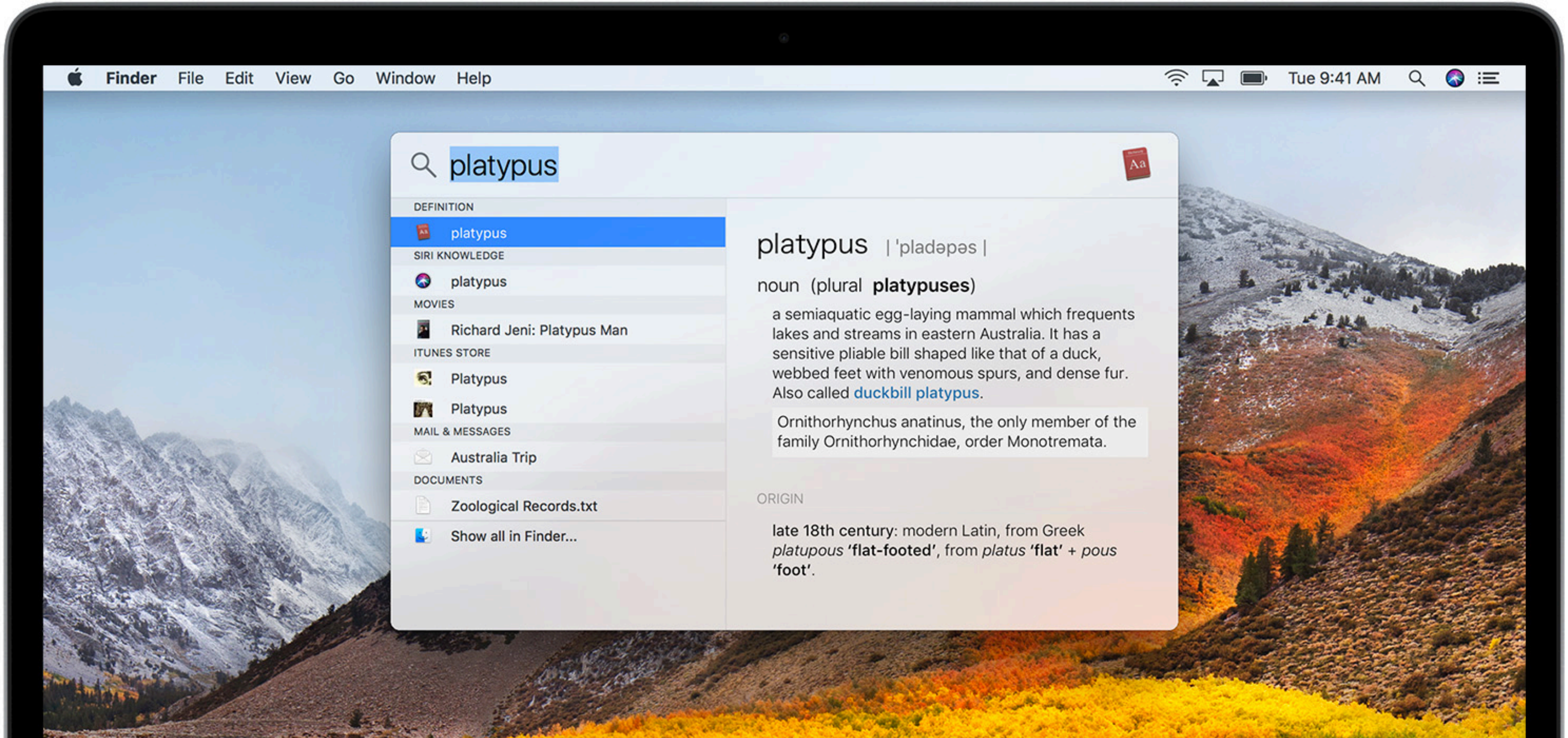
Background

Experiment Design

Results

Summary

WHAT IS SPOTLIGHT?



platypus | ˈpladəpəs |

noun (plural **platypuses**)

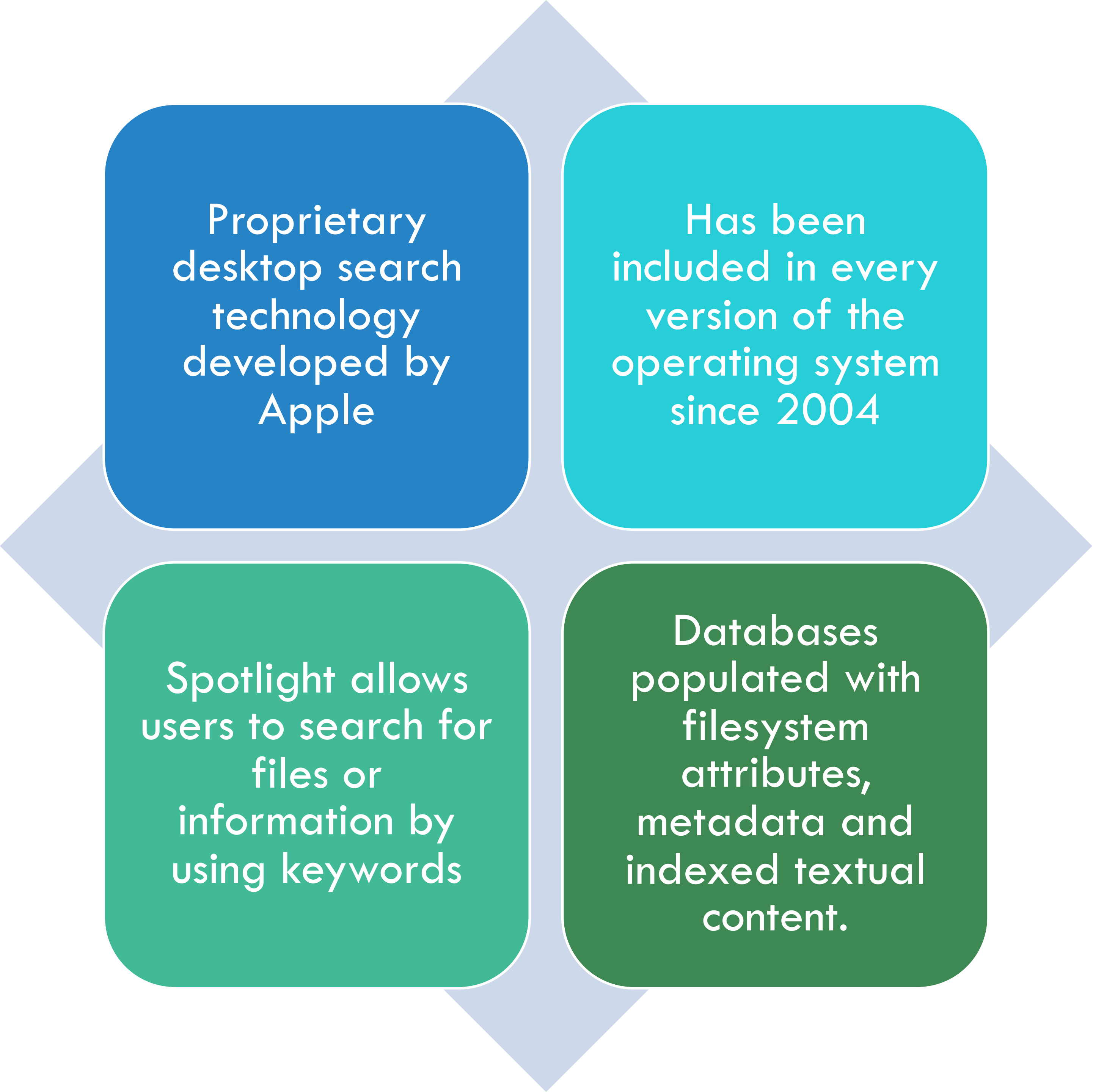
a semiaquatic egg-laying mammal which frequents lakes and streams in eastern Australia. It has a sensitive pliable bill shaped like that of a duck, webbed feet with venomous spurs, and dense fur. Also called duckbill platypus.

Ornithorhynchus anatinus, the only member of the family Ornithorhynchidae, order Monotremata.

ORIGIN

late 18th century: modern Latin, from Greek *platupous* **'flat-footed'**, from *platus* **'flat'** + *pous* **'foot'**.

WHAT IS SPOTLIGHT?



Proprietary desktop search technology developed by Apple

Has been included in every version of the operating system since 2004

Spotlight allows users to search for files or information by using keywords

Databases populated with filesystem attributes, metadata and indexed textual content.

WHY THIS SUBJECT

Over the last 15 years, the popularity and Market share of Apple devices and computers has steadily increased

- Number of submissions to the lab has increased.
- Kudos applied to owning such products.

My interest in the subject began in 2011

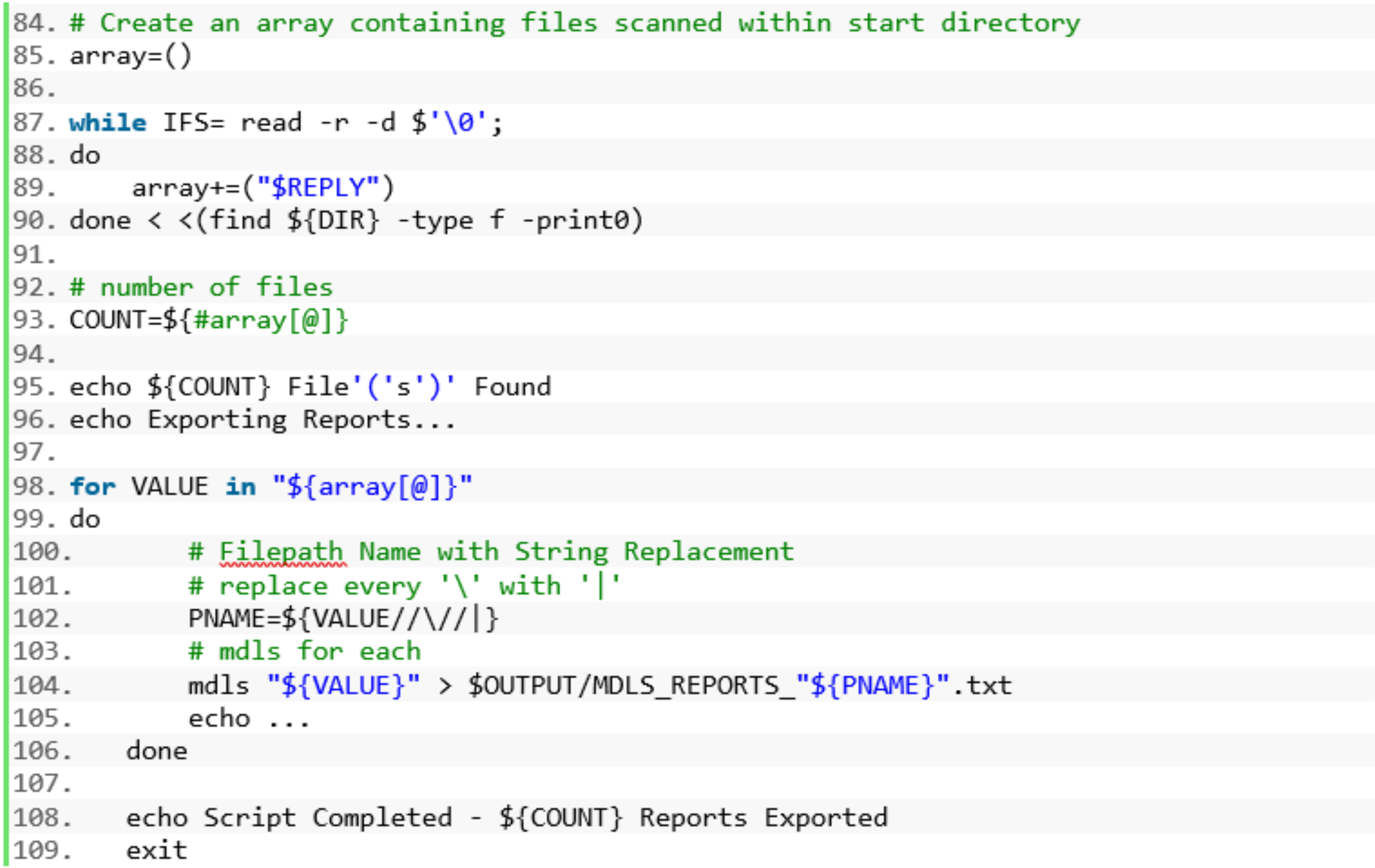
- Keyword hits within the Spotlight directories
- No tools available to parse the data
- No published research regarding the structure of the database

In 2013 504ensics announced Spotlight Inspector

- It disappeared shortly afterwards
- A tool to parse the database offline
- The structure was not revealed
- It was unknown if the tool was able to recover deleted records

By 2016, the method of examining Spotlight had not progressed:

- Use of Apple API to query extant records
- MDLS command using BASH



```
# Create an array containing files scanned within start directory
array=()

while IFS= read -r -d $'\0';
do
    array+=("$REPLY")
done < <(find ${DIR} -type f -print0)

# number of files
COUNT=${#array[@]}

echo ${COUNT} File'('s')' Found
echo Exporting Reports...

for VALUE in "${array[@]}"
do
        # Filepath Name with String Replacement
        # replace every '\' with '|'
        PNAME=${VALUE//\//|}
        # mdls for each
        mdls "${VALUE}" > $OUTPUT/MDLS_REPORTS_"${PNAME}".txt
        echo ...
    done

    echo Script Completed - ${COUNT} Reports Exported
    exit
```

WHY THIS SUBJECT

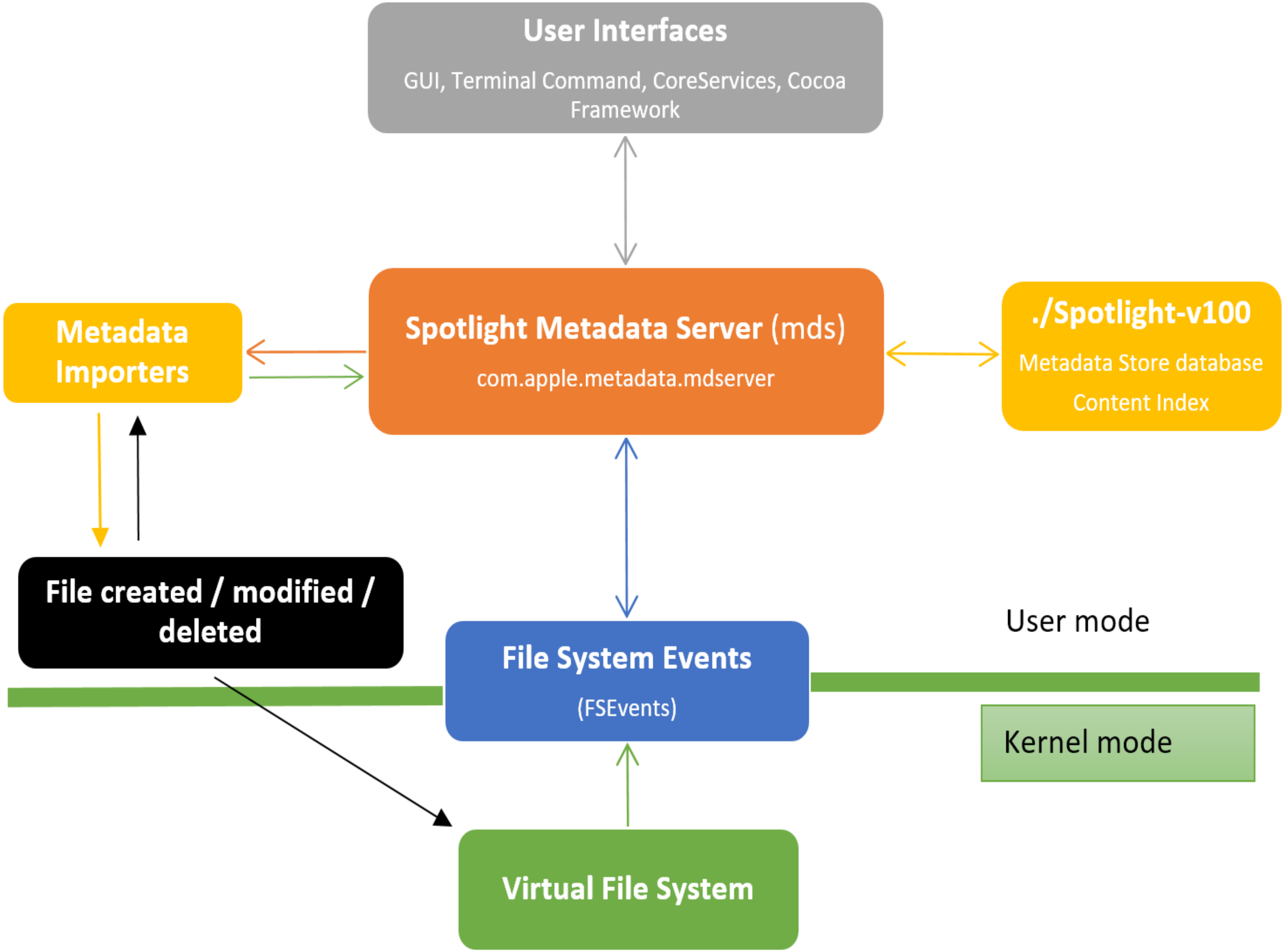
Metadata

- Is and always will be of great interest to digital forensic investigations
- To make the most if it, the method metadata is stored within files and data repositories must be:
  - **Understood**
  - **Tested** and
  - Shown to be to be **Reliable**

Parsing Data

- Parsing data from an unknown data format is inherently dangerous
- Incorrect linkage of records or incomplete data parsing can cause the data recovered to be:
  - **Misunderstood** and
  - Cause an **incorrect conclusion** to be drawn.

SPOTLIGHT COMPONENTS



FSEVENTS

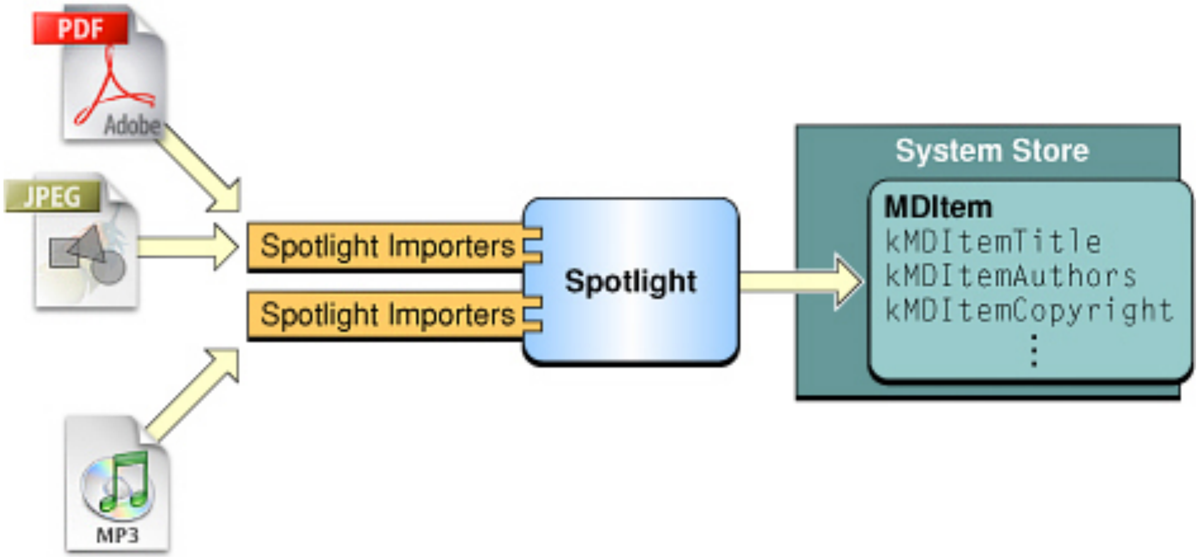
Event Type	Description
FSE_CREATE_FILE	File created
FSE_DELETE	File/directory removed/deleted
FSE_STAT_CHANGED	Change to status structure e.g. object's permission change
FSE_RENAME	File/directory renamed
FSE_CONTENT_MODIFIED	File content modified
FSE_EXCHANGE	Contents of two files were swapped through the exchangedata() system call
FSE_FINDER_INFO_CHANGED	File/directory Finder information changed e.g. label colour changed
FSE_CREATE_DIR	Folder created
FSE_CHOWN	Filesystem objects ownership changed
FSE_XATTR_MODIFIED	File/directory extended attributes modified
FSE_XATTR_REMOVED	File/directory extended attributes removed

METADATA IMPORTERS

Populate the content index store and metadata store database

A worker process, ‘mdworker’, launches the correct importer via a notification.

Each importer then extracts the metadata and contextual list of words, passing it back to the MDS for it to populate the stores



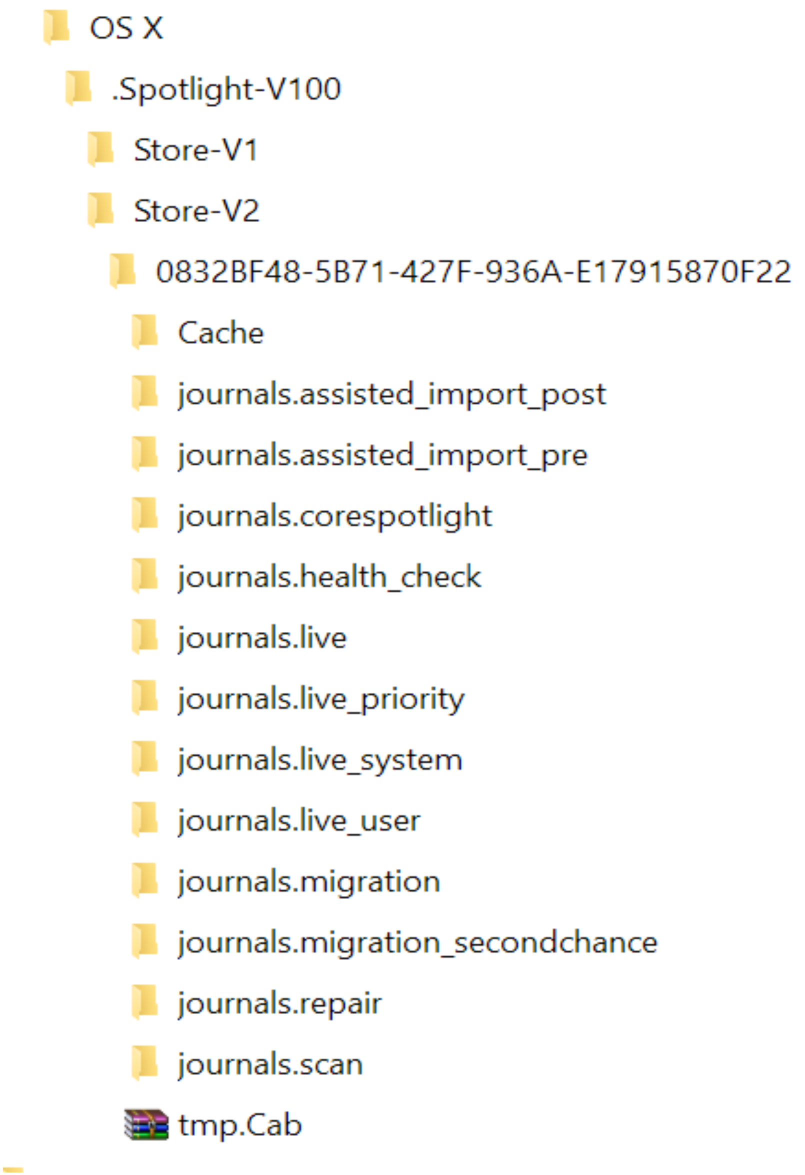
SPOTLIGHT STORE ARTEFACTS

The Spotlight-V100 directory is located in the root of the volume contains the Spotlight store.

Stores are only created on volumes where the operating system had read/write permissions.

Its presence indicates that the volume has been indexed

The property list file: *VolumeConfiguration.plist* details the location of all the Spotlight stores being held on the local system

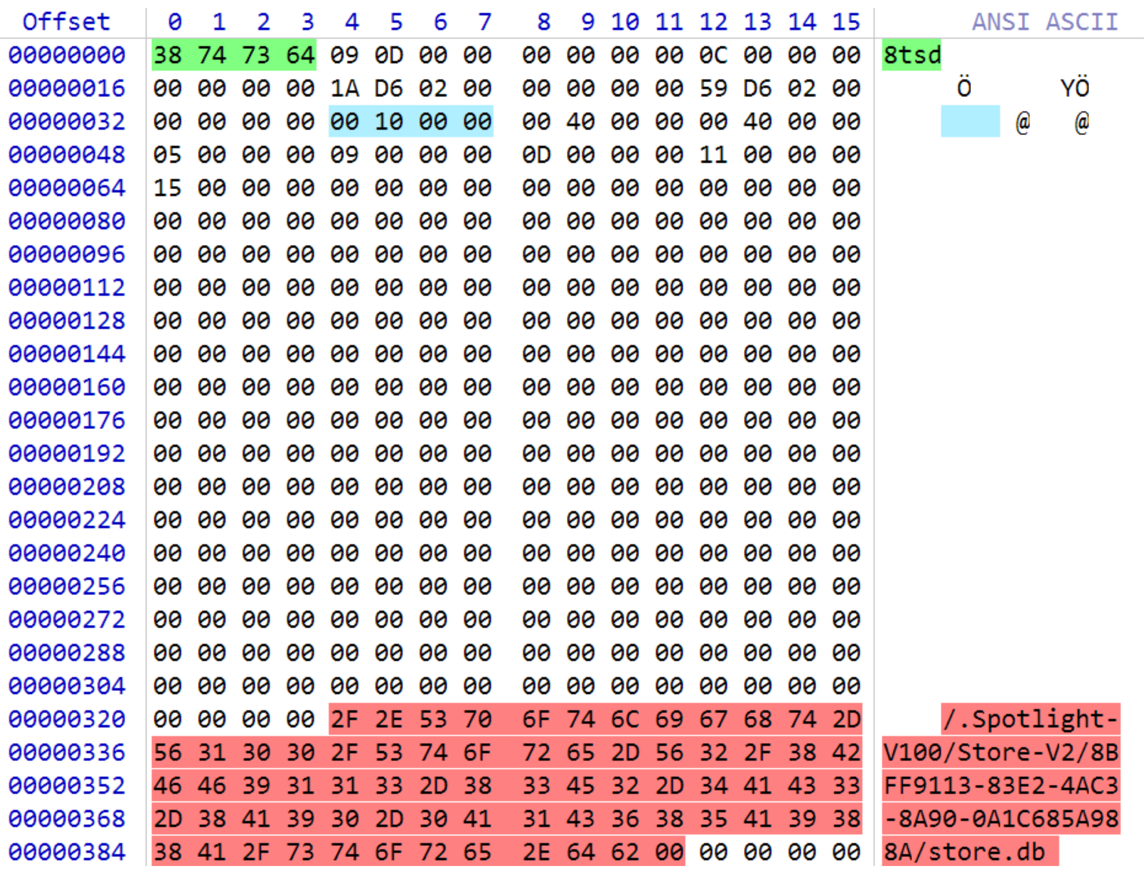


```
OS X
  .Spotlight-V100
    Store-V1
    Store-V2
      0832BF48-5B71-427F-936A-E17915870F22
        Cache
        journals.assisted_import_post
        journals.assisted_import_pre
        journals.corespotlight
        journals.health_check
        journals.live
        journals.live_priority
        journals.live_system
        journals.live_user
        journals.migration
        journals.migration_secondchance
        journals.repair
        journals.scan
        tmp.Cab
```

METADATA STORE DATABASE (STORE.DB)

Used as Spotlight’s central database repository for storing extracted metadata values.

A proprietary database whose structure is not published or described.



```
  Offset    0  1  2  3  4  5  6  7   8  9 10 11 12 13 14 15      ANSI ASCII
00000000   38 74 73 64 09 0D 00 00  00 00 00 00 0C 00 00 00   8tsd
00000016   00 00 00 00 1A D6 02 00  00 00 00 00 59 D6 02 00        Ö      YÖ
00000032   00 00 00 00 00 10 00 00  00 40 00 00 00 40 00 00            @   @
00000048   05 00 00 00 09 00 00 00  0D 00 00 00 11 00 00 00
00000064   15 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000080   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000096   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000112   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000128   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000144   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000160   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000176   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000192   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000208   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000224   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000240   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000256   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000272   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000288   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000304   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000320   00 00 00 00 2F 2E 53 70  6F 74 6C 69 67 68 74 2D       /.Spotlight-
00000336   56 31 30 30 2F 53 74 6F  72 65 2D 56 32 2F 38 42   V100/Store-V2/8B
00000352   46 46 39 31 31 33 2D 38  33 45 32 2D 34 41 43 33   FF9113-83E2-4AC3
00000368   2D 38 41 39 30 2D 30 41  31 43 36 38 35 41 39 38   -8A90-0A1C685A98
00000384   38 41 2F 73 74 6F 72 65  2E 64 62 00 00 00 00 00   8A/store.db
```

WHAT'S ALREADY OUT THERE?

Existing research has provided an understanding of the metadata attributes stored within the metadata store database (store.db).

Current methods of extracting information from the database make use of the Spotlight technology itself, with queries sent from an investigation workstation to perform live searches.

This approach has enabled forensic analysts to extract metadata records for extant files but not for deleted files.

No research was found that:

- **Proves deleted records are recoverable either directly from the metadata store or from unallocated space on the filesystem.**
- **Describes the structure of the metadata store database (store.db).**

APPROACH TO THIS WORK

Establish the structure of the spotlight database

- In order to identify records related to files

Establish what happens to deleted records

- Are deleted records are recoverable from the spotlight database (store.db)
- How long are deleted records recoverable before being lost
- What happens if a user intentionally destroys the spotlight index
- Can deleted versions of the database or database pages be recovered within unallocated areas

EXPERIMENT DESIGN

Virtualise Mac OS X machines

Populate the file system and the spotlight database with identifiable files (file name, file size, metadata, dates and times)

Develop Scripts that enable me to process the database structures that:

automate the identification and carving of database structures (i.e. pages, headers, records)

Process compressed or encrypted content

Identify offsets/relative offsets for records/flags

Exploit the newly discovered database structure to

Locate deleted records within the database and parse the information from them

Search for deleted databases within unallocated clusters

EXPERIMENT DESIGN

Experiment	Summary
01	Persistence of metadata records within the metadata store and unused space on the filesystem
02	Persistence of records **on mounted volumes**
03	Persistence of records **on mounted volumes that are shared across two operating systems**
04	Persistence of records **when the Spotlight indices are deleted using the mdutil terminal command**
05	Persistence records **when the Spotlight indices are deleted via the Spotlight management GUI interface**
06	Persistence of records **when the Spotlight indices are deleted using the mdutil terminal command and re-populated with ever increasing number of files.**
07	Creation of metadata records for **the purposes of reverse engineering the metadata store structure**
08	Persistence of records **when the operating system is upgraded (minor/major)**
09	Persistence of records **within the unused space of ten casework forensic images**

EXPERIMENT RESULTS

STORE.DB
DATABASE PAGES

Three main types of database pages identified

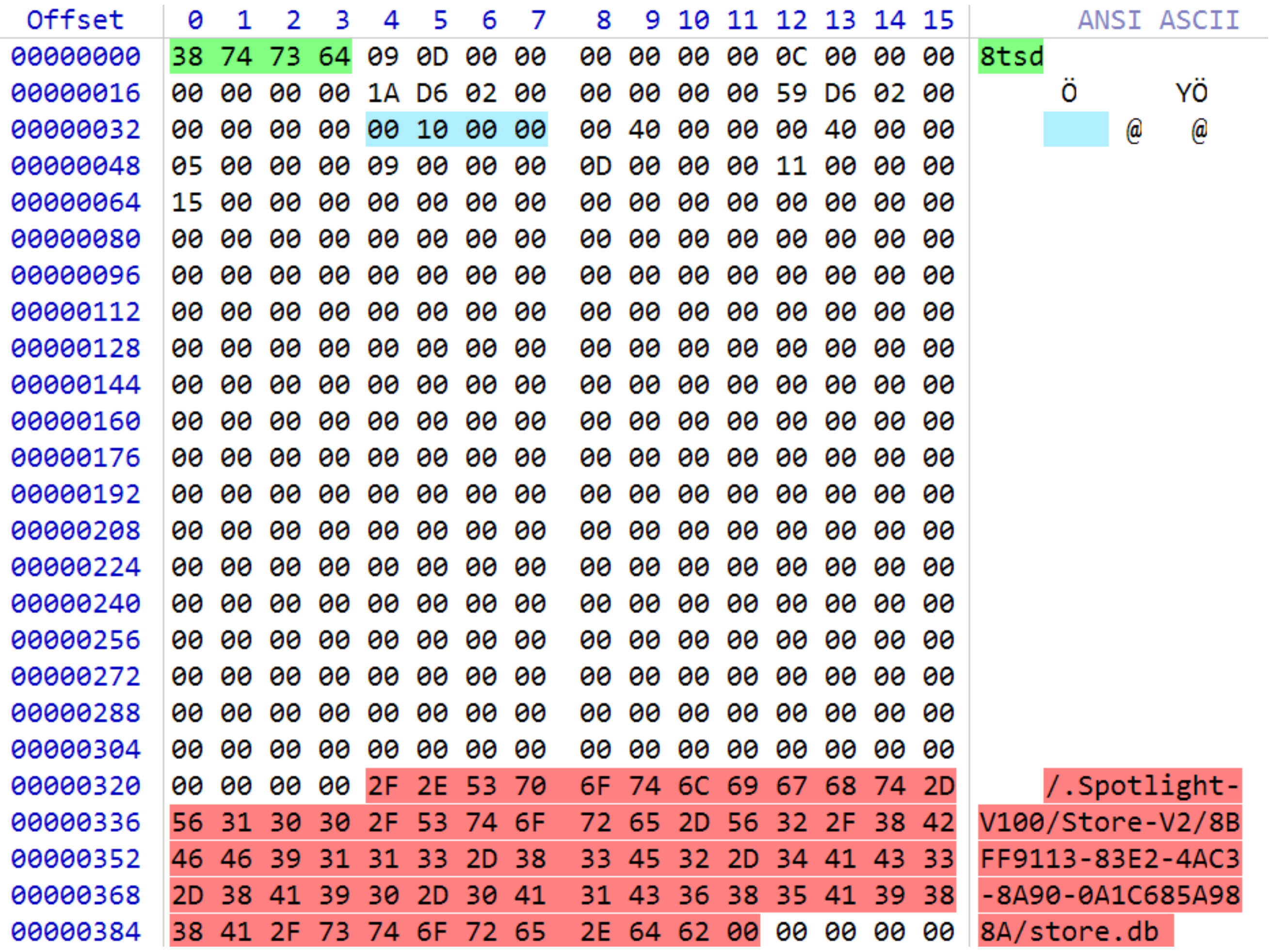
Each identifiable by the 4-byte signature located at relative offset 0.

Page Header String (Hexadecimal)	Offset	Size (bytes)	String ASCII	Page Type
38 74 73 64	00	4	8tsd	Header Page
32 6D 62 64	00	4	2mbd	Map Page
32 70 62 64	00	4	2pbd	Data Page

STORE.DB HEADER PAGE

The first page encountered is identified as the database header.

In experiments, this page has consistently been 4096 bytes in length.



```
Offset     0  1  2  3  4  5  6  7   8  9 10 11 12 13 14 15   ANSI ASCII
00000000  38 74 73 64 09 0D 00 00  00 00 00 00 0C 00 00 00  8tsd
00000016  00 00 00 00 1A D6 02 00  00 00 00 00 59 D6 02 00       Ö      YÖ
00000032  00 00 00 00 00 10 00 00  00 40 00 00 00 40 00 00           @   @
00000048  05 00 00 00 09 00 00 00  0D 00 00 00 11 00 00 00
00000064  15 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000096  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000112  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000128  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000144  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000160  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000176  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000192  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000208  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000224  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000240  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000256  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000272  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000288  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000304  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000320  00 00 00 00 2F 2E 53 70  6F 74 6C 69 67 68 74 2D      /.Spotlight-
00000336  56 31 30 30 2F 53 74 6F  72 65 2D 56 32 2F 38 42  V100/Store-V2/8B
00000352  46 46 39 31 31 33 2D 38  33 45 32 2D 34 41 43 33  FF9113-83E2-4AC3
00000368  2D 38 41 39 30 2D 30 41  31 43 36 38 35 41 39 38  -8A90-0A1C685A98
00000384  38 41 2F 73 74 6F 72 65  2E 64 62 00 00 00 00 00  8A/store.db
```

Offset	Bytes	Description
00	4 bytes	Header String recorded as 8tsd
36	4 bytes	Database Header Page Size
324	variable	Path to store.db on volume. String null terminated

STORE.DB MAP PAGE

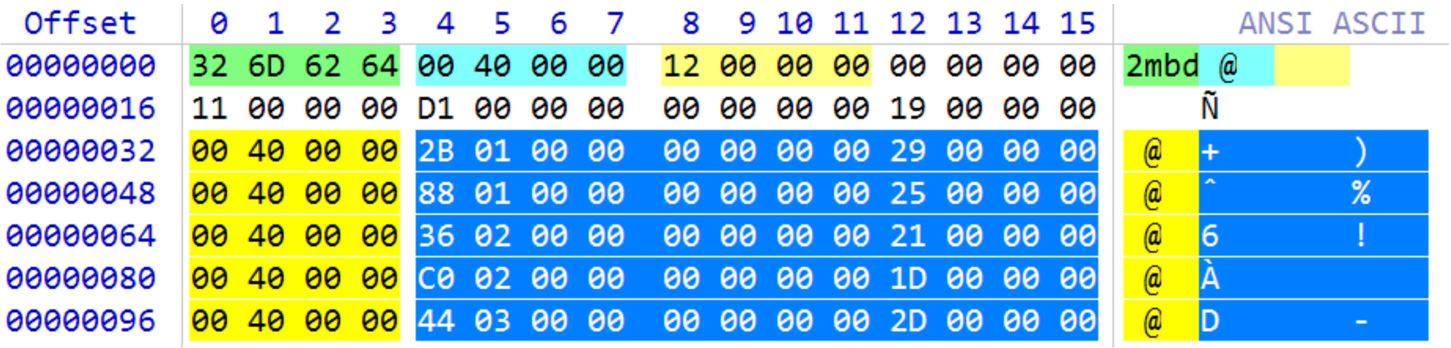
The second type of page encountered has been identified as the database map.

This describes each data page encountered within the database

The first 32 bytes provide information regarding the page.

Starting at offset 32, each data page is described by 16 bytes.

The first 4 bytes declare the size of each data page with the remaining 12 bytes unknown.



```
 Offset    0  1  2  3   4  5  6  7   8  9 10 11 12 13 14 15    ANSI ASCII
00000000  32 6D 62 64  00 40 00 00  12 00 00 00 00 00 00 00   2mbd @
00000016  11 00 00 00  D1 00 00 00  00 00 00 00 19 00 00 00       Ñ
00000032  00 40 00 00  2B 01 00 00  00 00 00 00 29 00 00 00    @  +       )
00000048  00 40 00 00  88 01 00 00  00 00 00 00 25 00 00 00    @  ˆ       %
00000064  00 40 00 00  36 02 00 00  00 00 00 00 21 00 00 00    @  6       !
00000080  00 40 00 00  C0 02 00 00  00 00 00 00 1D 00 00 00    @  À
00000096  00 40 00 00  44 03 00 00  00 00 00 00 2D 00 00 00    @  D       -
```

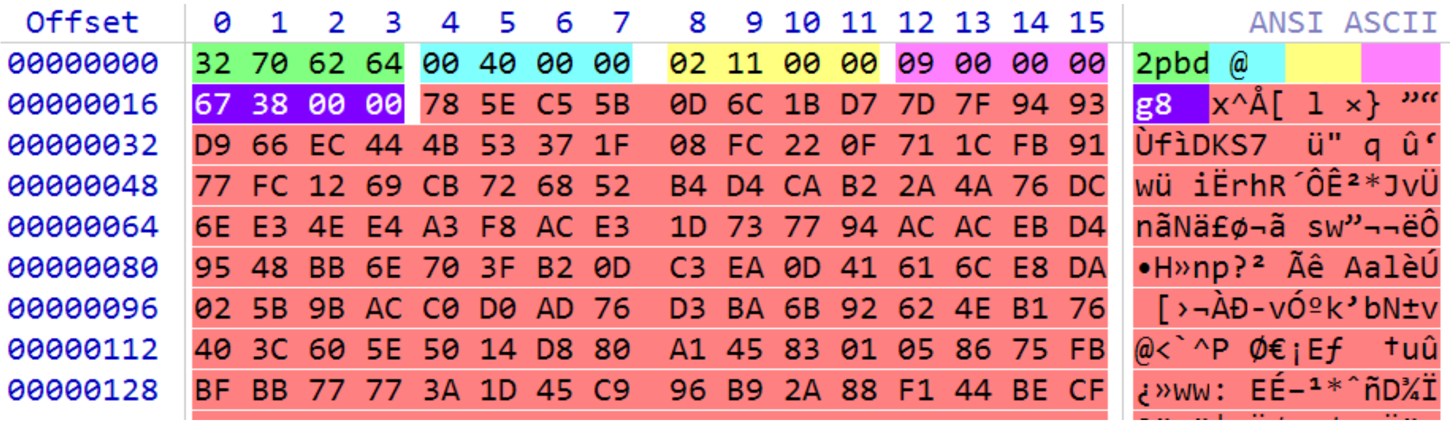
Offset	Bytes	Description
00	4 bytes	Header String recorded as 2mbd
04	4 bytes	Page Size always observed as 16384 bytes in length
08	4 bytes	Number of pages contained within the map
32	16 bytes	Start of the page map entries.

STORE.DB DATA PAGE

- These pages contain different types of data described further in the paper but in this talk we will concentrate on the metadata pages

- The header of the data pages are identical

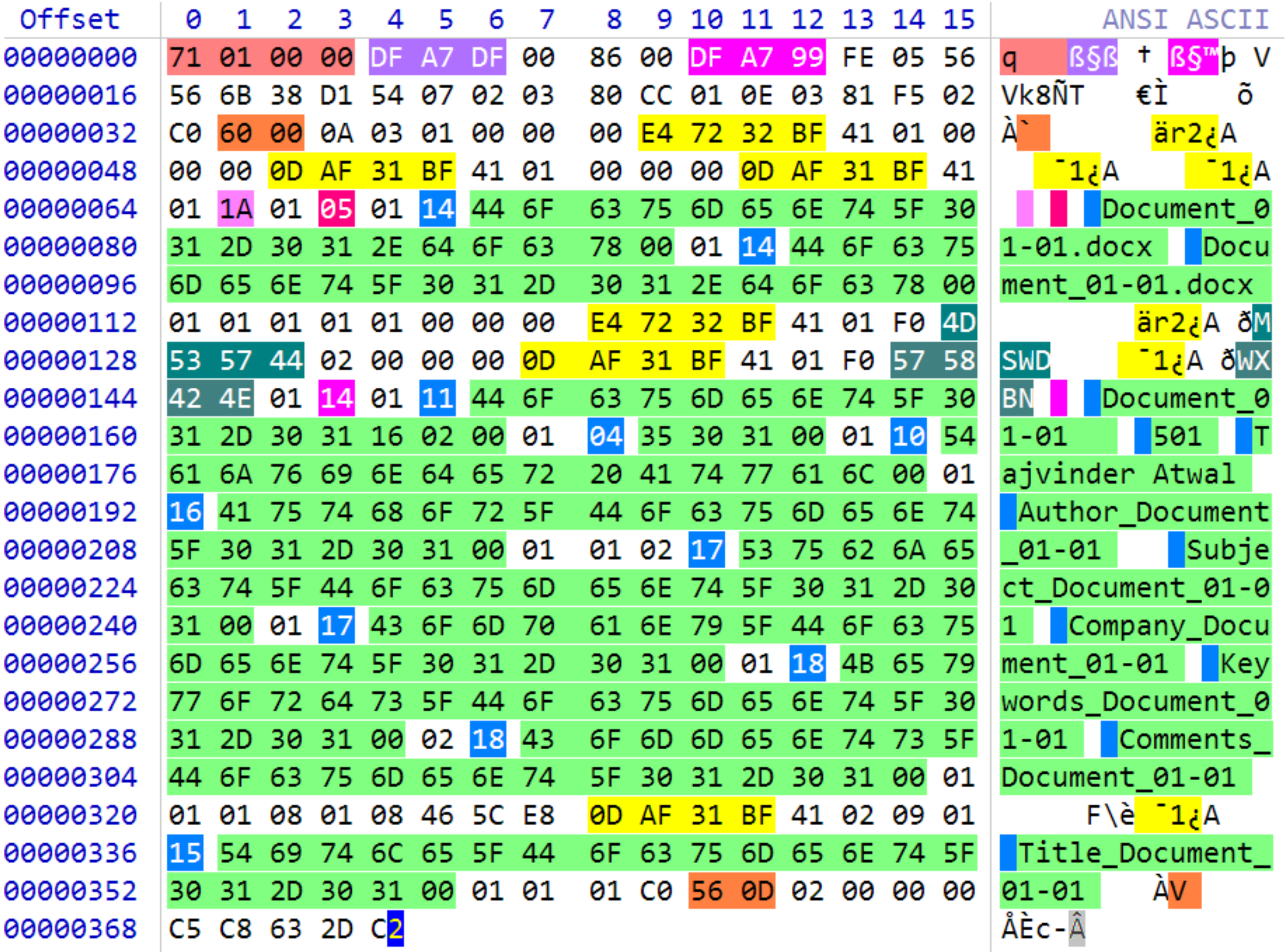
- The key records used for parsing all data pages are found within the first 20 bytes



```
Offset    0  1  2  3  4  5  6  7   8  9 10 11 12 13 14 15   ANSI ASCII
00000000 32 70 62 64 00 40 00 00  02 11 00 00 09 00 00 00  2pbd @
00000016 67 38 00 00 78 5E C5 5B  0D 6C 1B D7 7D 7F 94 93  g8  x^Å[ l ×} ”“
00000032 D9 66 EC 44 4B 53 37 1F  08 FC 22 0F 71 1C FB 91  ÙfìDKS7  ü" q û‘
00000048 77 FC 12 69 CB 72 68 52  B4 D4 CA B2 2A 4A 76 DC  wü iËrhR´ÔÊ²*JvÜ
00000064 6E E3 4E E4 A3 F8 AC E3  1D 73 77 94 AC AC EB D4  nãNä£ø¬ã sw”¬¬ëÔ
00000080 95 48 BB 6E 70 3F B2 0D  C3 EA 0D 41 61 6C E8 DA  •H»np?² Ãê AalèÚ
00000096 02 5B 9B AC C0 D0 AD 76  D3 BA 6B 92 62 4E B1 76   [›¬ÀÐ-vÓºk’bN±v
00000112 40 3C 60 5E 50 14 D8 80  A1 45 83 01 05 86 75 FB  @<`^P Ø€¡Eƒ  †uû
00000128 BF BB 77 77 3A 1D 45 C9  96 B9 2A 88 F1 44 BE CF  ¿»ww: EÉ–¹*ˆñD¾Ï
```

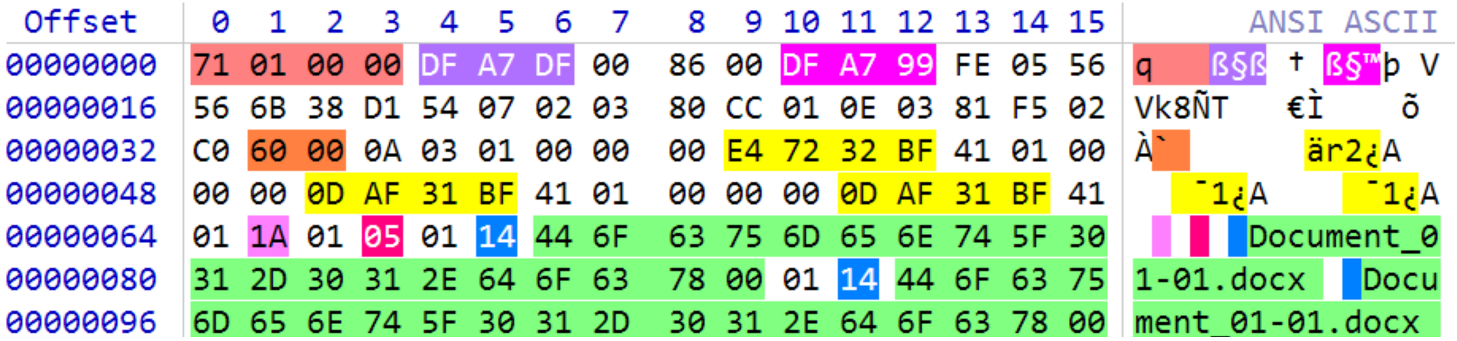
Offset	Bytes	Description
00	4 bytes	Header String recorded as 2pbd
04	4 bytes	Page Size (Physical)
08	4 bytes	Size (Allocated)
12	4 bytes	Record page sub-type
16	4 bytes	Size (Uncompressed)
20	variable	Data

STORE.DB DATA PAGE (DECOMPRESSED)



```
Offset      0  1  2  3  4  5  6  7   8  9 10 11 12 13 14 15   ANSI ASCII
00000000   71 01 00 00 DF A7 DF 00  86 00 DF A7 99 FE 05 56   q   ß§ß † ß§™þ V
00000016   56 6B 38 D1 54 07 02 03  80 CC 01 0E 03 81 F5 02   Vk8ÑT   €Ì    õ
00000032   C0 60 00 0A 03 01 00 00  00 E4 72 32 BF 41 01 00   À`        är2¿A
00000048   00 00 0D AF 31 BF 41 01  00 00 00 0D AF 31 BF 41     ¯1¿A     ¯1¿A
00000064   01 1A 01 05 01 14 44 6F  63 75 6D 65 6E 74 5F 30         Document_0
00000080   31 2D 30 31 2E 64 6F 63  78 00 01 14 44 6F 63 75   1-01.docx   Docu
00000096   6D 65 6E 74 5F 30 31 2D  30 31 2E 64 6F 63 78 00   ment_01-01.docx
00000112   01 01 01 01 01 00 00 00  E4 72 32 BF 41 01 F0 4D           är2¿A ðM
00000128   53 57 44 02 00 00 00 0D  AF 31 BF 41 01 F0 57 58   SWD     ¯1¿A ðWX
00000144   42 4E 01 14 01 11 44 6F  63 75 6D 65 6E 74 5F 30   BN    Document_0
00000160   31 2D 30 31 16 02 00 01  04 35 30 31 00 01 10 54   1-01     501   T
00000176   61 6A 76 69 6E 64 65 72  20 41 74 77 61 6C 00 01   ajvinder Atwal
00000192   16 41 75 74 68 6F 72 5F  44 6F 63 75 6D 65 6E 74    Author_Document
00000208   5F 30 31 2D 30 31 00 01  01 02 17 53 75 62 6A 65   _01-01     Subje
00000224   63 74 5F 44 6F 63 75 6D  65 6E 74 5F 30 31 2D 30   ct_Document_01-0
00000240   31 00 01 17 43 6F 6D 70  61 6E 79 5F 44 6F 63 75   1   Company_Docu
00000256   6D 65 6E 74 5F 30 31 2D  30 31 00 01 18 4B 65 79   ment_01-01   Key
00000272   77 6F 72 64 73 5F 44 6F  63 75 6D 65 6E 74 5F 30   words_Document_0
00000288   31 2D 30 31 00 02 18 43  6F 6D 6D 65 6E 74 73 5F   1-01   Comments_
00000304   44 6F 63 75 6D 65 6E 74  5F 30 31 2D 30 31 00 01   Document_01-01
00000320   01 01 08 01 08 46 5C E8  0D AF 31 BF 41 02 09 01        F\è ¯1¿A
00000336   15 54 69 74 6C 65 5F 44  6F 63 75 6D 65 6E 74 5F    Title_Document_
00000352   30 31 2D 30 31 00 01 01  01 C0 56 0D 02 00 00 00   01-01     ÀV
00000368   C5 C8 63 2D C2                                     ÅÈc-Â
```

STORE.DB DATA PAGE (DECOMPRESSED)



```
Offset    0  1  2  3  4  5  6  7   8  9 10 11 12 13 14 15   ANSI ASCII
00000000 71 01 00 00 DF A7 DF 00  86 00 DF A7 99 FE 05 56  q   ßß§ß † ß§™þ V
00000016 56 6B 38 D1 54 07 02 03  80 CC 01 0E 03 81 F5 02  Vk8ÑT   €Ì    õ 
00000032 C0 60 00 0A 03 01 00 00  00 E4 72 32 BF 41 01 00  À`       är2¿A  
00000048 00 00 0D AF 31 BF 41 01  00 00 00 0D AF 31 BF 41     ¯1¿A     ¯1¿A
00000064 01 1A 01 05 01 14 44 6F  63 75 6D 65 6E 74 5F 30        Document_0
00000080 31 2D 30 31 2E 64 6F 63  78 00 01 14 44 6F 63 75  1-01.docx   Docu
00000096 6D 65 6E 74 5F 30 31 2D  30 31 2E 64 6F 63 78 00  ment_01-01.docx 
```

Of particular interest, is the existence of the Catalog Node ID (CNID) and Parent CNID markers found within the records.

- It is similar in structure to a network database
- Each record maintains a relationship with its parent
- It allows a hierarchical arrangement of records to be built.

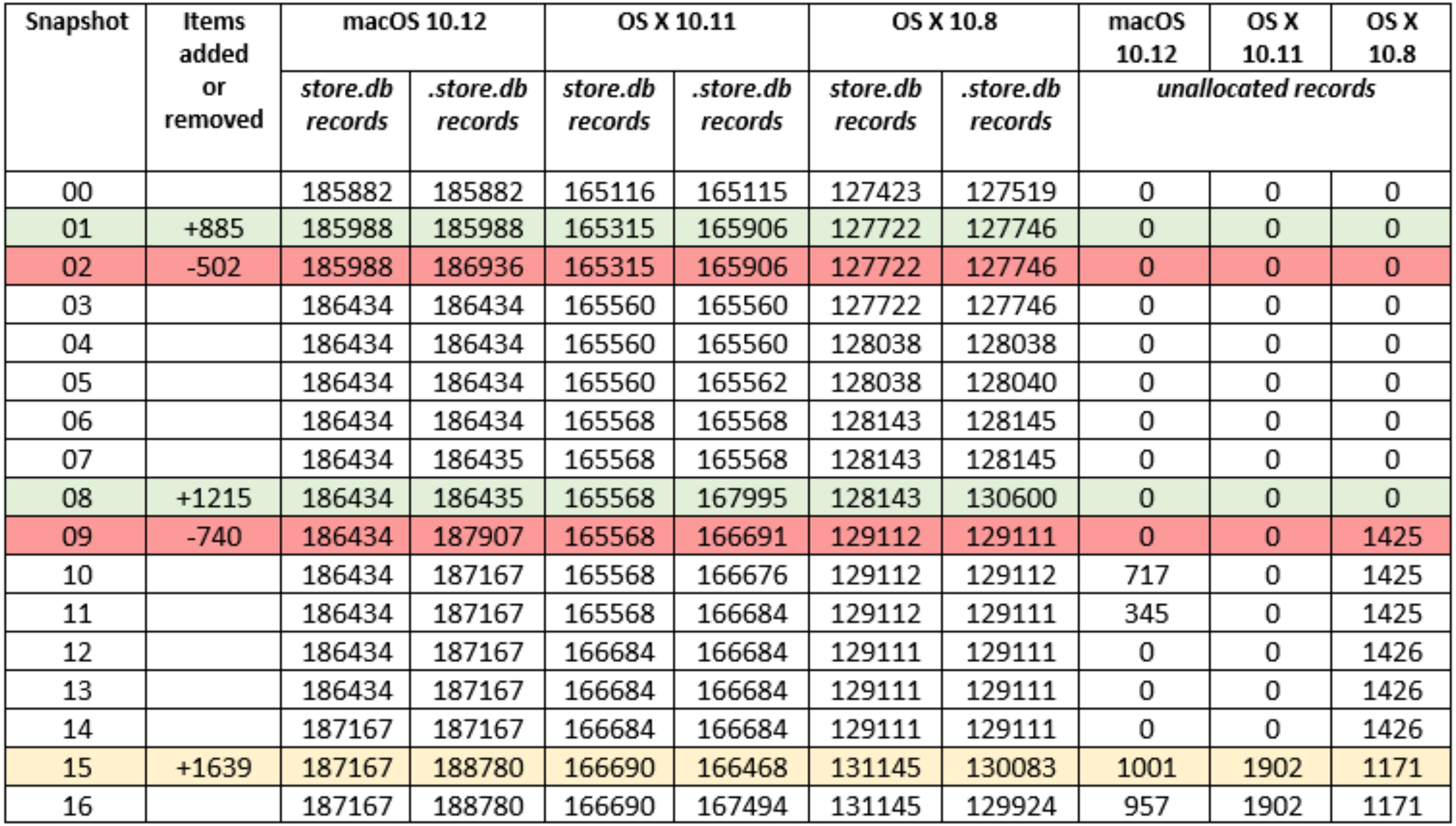
The Catalog Node ID is used by the HFS+ filesystem to uniquely identify each file/folder on the system.

- An important feature is that they are not reused until exhausted
- New files created on an HFS+ filesystem are given the next consecutive CNID, even if an earlier one has been made available because of a deletion.

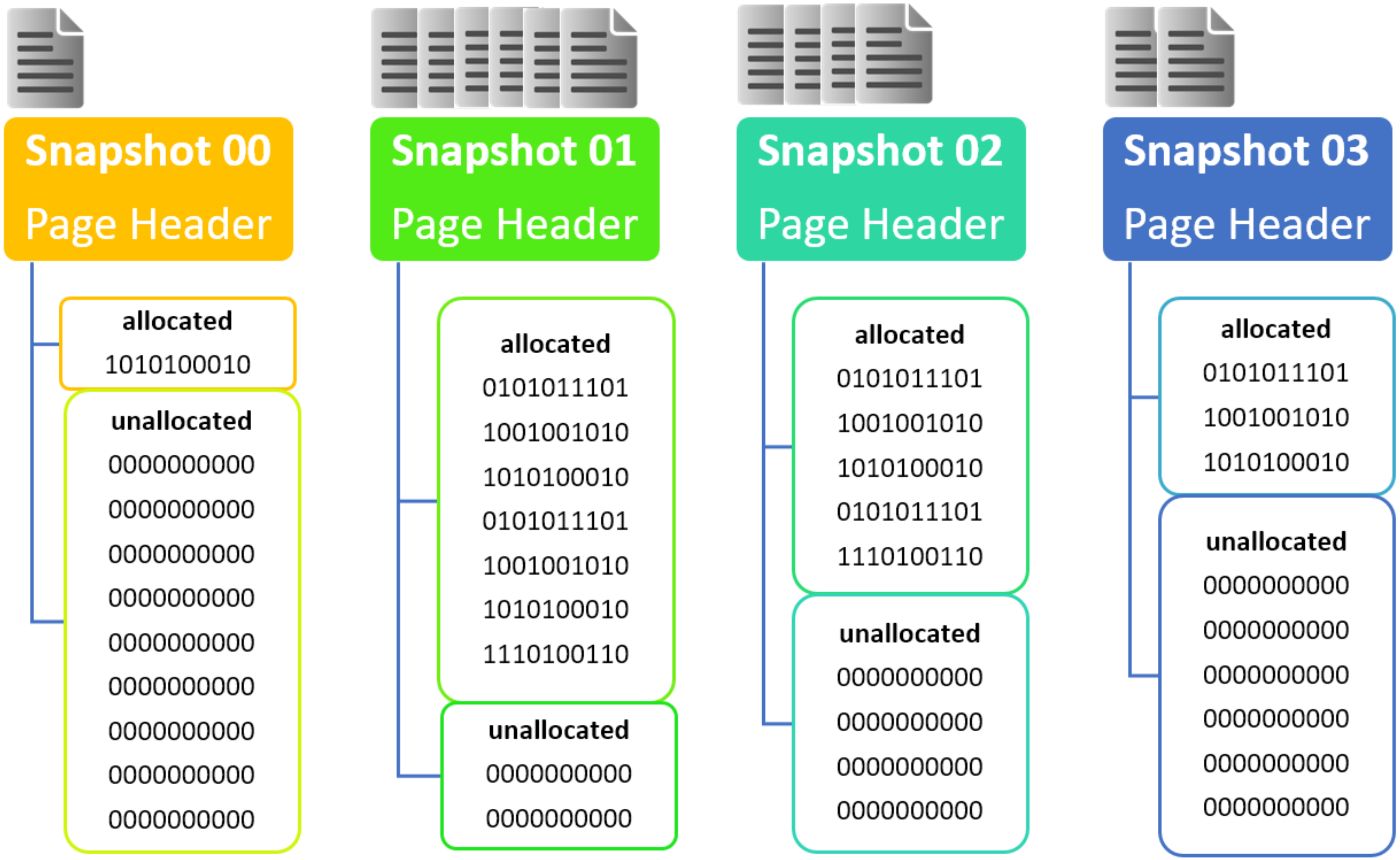
Why is this important?

- **We can use it to identify deleted files, files that no longer exist within the current `$Catalog` main index found within HFS+**
- **We can re-create directory structures**

EXPERIMENT 1



Snapshot	Items added or removed	macOS 10.12		OS X 10.11		OS X 10.8		macOS 10.12	OS X 10.11	OS X 10.8
		store.db records	*.store.db records*	*store.db records*	*.store.db records*	*store.db records*	*.store.db records*	*unallocated records*		
00		185882	185882	165116	165115	127423	127519	0	0	0
01	+885	185988	185988	165315	165906	127722	127746	0	0	0
02	-502	185988	186936	165315	165906	127722	127746	0	0	0
03		186434	186434	165560	165560	127722	127746	0	0	0
04		186434	186434	165560	165560	128038	128038	0	0	0
05		186434	186434	165560	165562	128038	128040	0	0	0
06		186434	186434	165568	165568	128143	128145	0	0	0
07		186434	186435	165568	165568	128143	128145	0	0	0
08	+1215	186434	186435	165568	167995	128143	130600	0	0	0
09	-740	186434	187907	165568	166691	129112	129111	0	0	1425
10		186434	187167	165568	166676	129112	129112	717	0	1425
11		186434	187167	165568	166684	129112	129111	345	0	1425
12		186434	187167	166684	166684	129111	129111	0	0	1426
13		186434	187167	166684	166684	129111	129111	0	0	1426
14		187167	187167	166684	166684	129111	129111	0	0	1426
15	+1639	187167	188780	166690	166468	131145	130083	1001	1902	1171
16		187167	188780	166690	167494	131145	129924	957	1902	1171



Snapshot 00

Page Header

allocated

1010100010

unallocated

0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000

Snapshot 01

Page Header

allocated

0101011101
1001001010
1010100010
0101011101
1001001010
1010100010
1110100110

unallocated

0000000000
0000000000

Snapshot 02

Page Header

allocated

0101011101
1001001010
1010100010
0101011101
1110100110

unallocated

0000000000
0000000000
0000000000
0000000000

Snapshot 03

Page Header

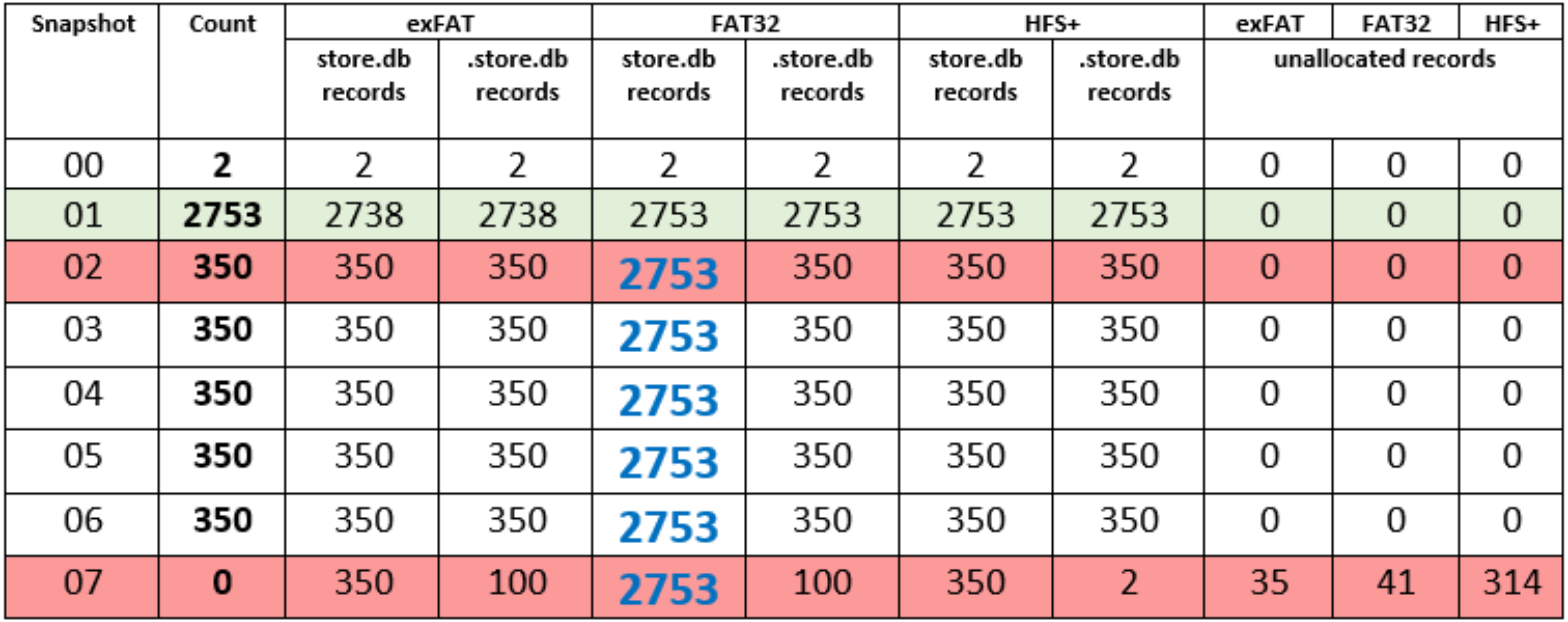
allocated

0101011101
1001001010
1010100010

unallocated

0000000000
0000000000
0000000000
0000000000
0000000000
0000000000

EXPERIMENT 2

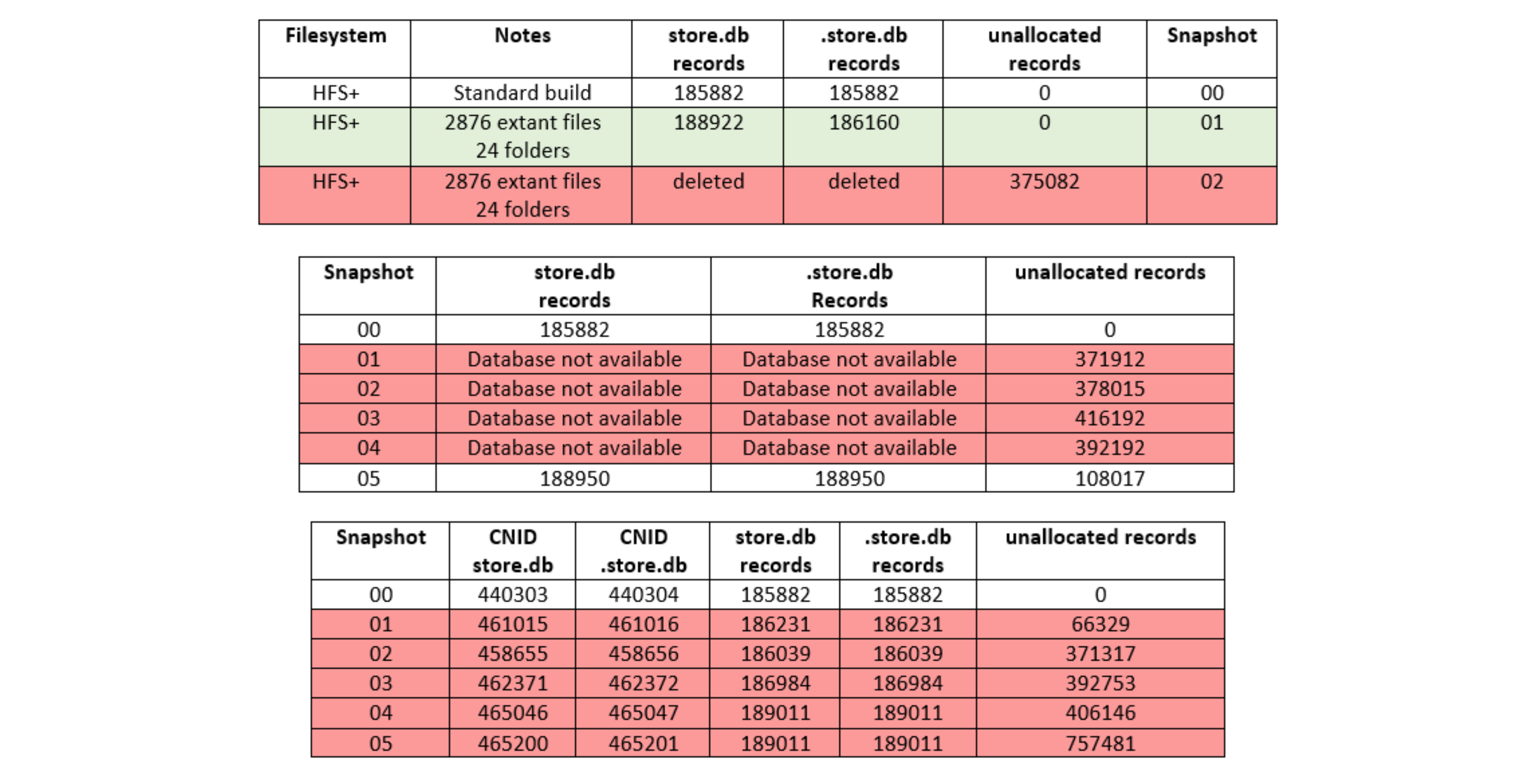


Snapshot	Count	exFAT		FAT32		HFS+		exFAT	FAT32	HFS+
		store.db records	.store.db records	store.db records	.store.db records	store.db records	.store.db records	unallocated records		
00	**2**	2	2	2	2	2	2	0	0	0
01	**2753**	2738	2738	2753	2753	2753	2753	0	0	0
02	**350**	350	350	**2753**	350	350	350	0	0	0
03	**350**	350	350	**2753**	350	350	350	0	0	0
04	**350**	350	350	**2753**	350	350	350	0	0	0
05	**350**	350	350	**2753**	350	350	350	0	0	0
06	**350**	350	350	**2753**	350	350	350	0	0	0
07	**0**	350	100	**2753**	100	350	2	35	41	314

EXPERIMENT 3

Snapshot	Count	exFAT		FAT32		exFAT	FAT32
		store.db records	.store.db records	store.db records	.store.db records	unallocated records	
00	2	2	2	2	2	0	0
01	2753	7	1526	7	1480	0	0
02	377	483	483	392	392	107	380
03	377	483	483	392	392	107	380
04	2	483	481	392	390	107	380
05	157	125	125	24	24	107	778
06	157	130	130	24	24	107	778

EXPERIMENTS 4, 5 AND 6

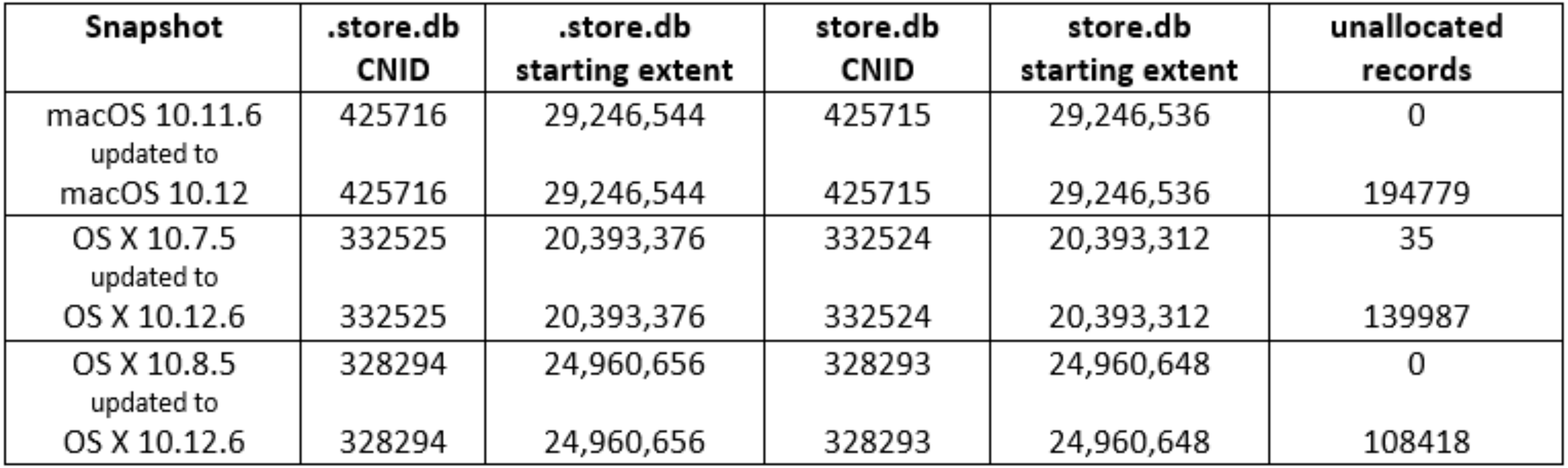


Filesystem	Notes	store.db records	.store.db records	unallocated records	Snapshot
HFS+	Standard build	185882	185882	0	00
HFS+	2876 extant files 24 folders	188922	186160	0	01
HFS+	2876 extant files 24 folders	deleted	deleted	375082	02

Snapshot	store.db records	.store.db Records	unallocated records
00	185882	185882	0
01	Database not available	Database not available	371912
02	Database not available	Database not available	378015
03	Database not available	Database not available	416192
04	Database not available	Database not available	392192
05	188950	188950	108017

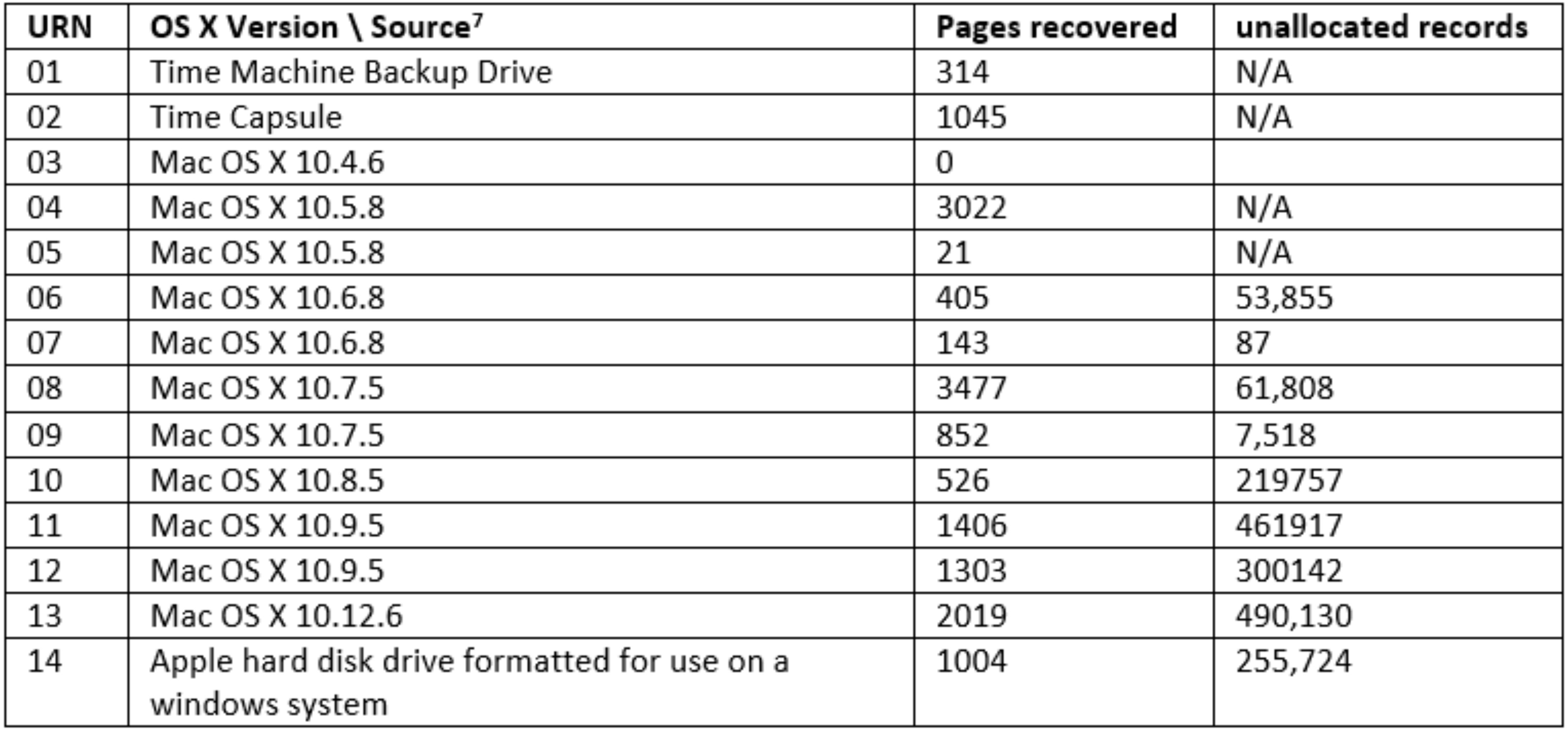
Snapshot	CNID store.db	CNID .store.db	store.db records	.store.db records	unallocated records
00	440303	440304	185882	185882	0
01	461015	461016	186231	186231	66329
02	458655	458656	186039	186039	371317
03	462371	462372	186984	186984	392753
04	465046	465047	189011	189011	406146
05	465200	465201	189011	189011	757481

EXPERIMENT 8



Snapshot	.store.db CNID	.store.db starting extent	store.db CNID	store.db starting extent	unallocated records
macOS 10.11.6	425716	29,246,544	425715	29,246,536	0
updated to					
macOS 10.12	425716	29,246,544	425715	29,246,536	194779
OS X 10.7.5	332525	20,393,376	332524	20,393,312	35
updated to					
OS X 10.12.6	332525	20,393,376	332524	20,393,312	139987
OS X 10.8.5	328294	24,960,656	328293	24,960,648	0
updated to					
OS X 10.12.6	328294	24,960,656	328293	24,960,648	108418

EXPERIMENT 9



URN	OS X Version \ Source[7]	Pages recovered	unallocated records
01	Time Machine Backup Drive	314	N/A
02	Time Capsule	1045	N/A
03	Mac OS X 10.4.6	0	
04	Mac OS X 10.5.8	3022	N/A
05	Mac OS X 10.5.8	21	N/A
06	Mac OS X 10.6.8	405	53,855
07	Mac OS X 10.6.8	143	87
08	Mac OS X 10.7.5	3477	61,808
09	Mac OS X 10.7.5	852	7,518
10	Mac OS X 10.8.5	526	219757
11	Mac OS X 10.9.5	1406	461917
12	Mac OS X 10.9.5	1303	300142
13	Mac OS X 10.12.6	2019	490,130
14	Apple hard disk drive formatted for use on a windows system	1004	255,724

OUTCOME

The structure of the metadata store database has been analysed and partially decoded

Experiments used to reveal that records persist for a period of time within one of the copies of the database

Once a record is deleted it is no longer recoverable from within the database but may still be recovered from the copy or from unallocated clusters.

Deleted pages from the database, are recoverable from unused space on the filesystem.

If the Operating system undergoes a major update it appears that a copy of the metadata store is created before being deleted.

These database pages can be recovered from unused space on the filesystem

If the Spotlight index is reset re-indexed or recreated, the whole metadata store is deleted. These database pages can be recovered from unused space on the filesystem

INTERESTED IN LEARNING MORE?

Digital Investigation

- Atwal, T. S., Scanlon, M. and Le-Khac, N-A. **Shining a Light on Spotlight: Leveraging Apple's Desktop Search Utility to Recover Deleted File Metadata on macOS**, *Digital Investigation, April 2019*
- Khatri, Yogesh. **Investigating spotlight internals to extract metadata.** Digital Investigation, March 2019.

GitHub

- https://github.com/tajatwal
- https://github.com/ydkhatri

SHINING A LIGHT ON SPOTLIGHT

Taj Atwal
taj@reduc.tech