A RESEARCH INITIATIVE OF TEAM FORENSICSWARE

Explore Outlook Data File Structure For Forensics

Contributor: Swati Upadhyay in Association with CCIRC Team
(http://www.ccirc.in)

Volume 1	Issue 2	Oct 10	2014

CCIRC: CYBER CRIME INVESTIGATION & RESEARCH CENTRE

Contributor: Swati Upadhyay in association with CCIRC team (http://www.ccirc.in)

Explore Outlook Data File Structure for Forensic

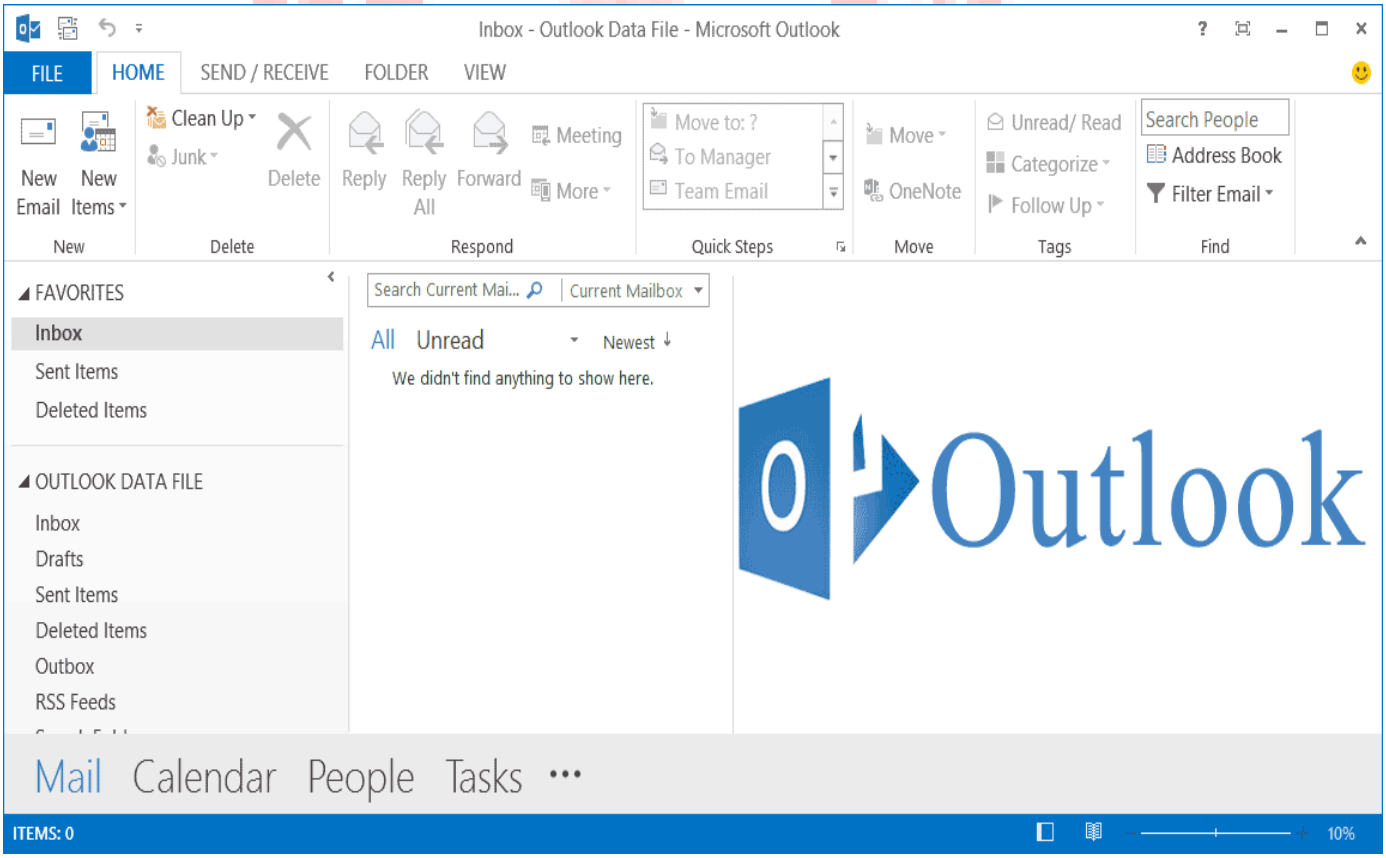


TABLE OF CONTENT:

ABSTRACT:

The Research Paper , "Explore Outlook Data File Structure for Forensic" enumerates the basic data file structure of Microsoft Outlook, which is a renowned Personal Information Manager and a desktop based email client. Information such as Outlook Data file Structure , Logical Architecture & its layers are explained from a digital forensics scope of view. It involes root level description of the PST file structure , how the data gets stored and traversed along with the logical methodology to depict them. The File structure faces instability issues and to overcome them some technical utilities that are introduced are; ScanPST.exe and SysTools Outlook Recovery.

INTRODUCTION:

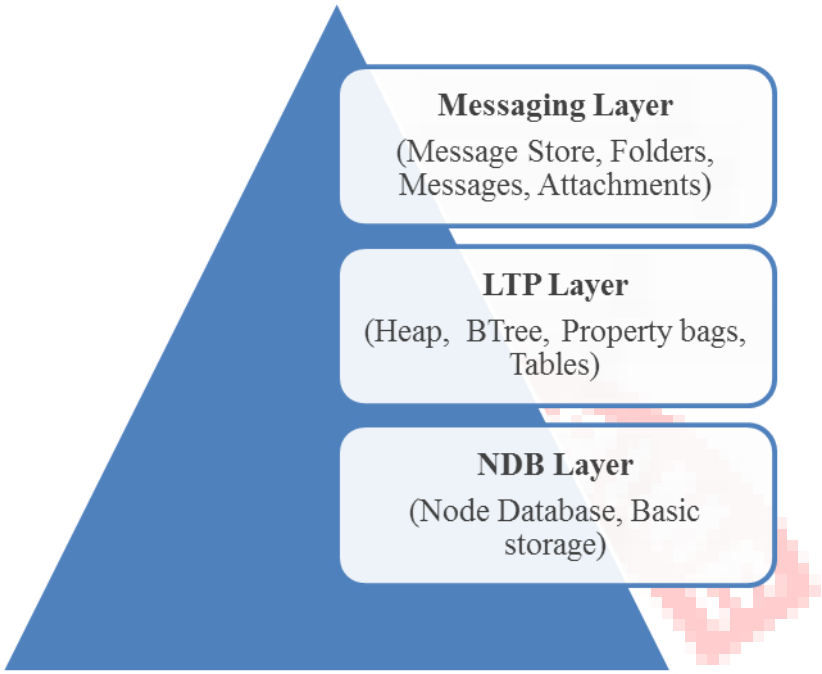
Microsoft Outlook, a part of the Microsoft Office suite is a Personal Information Manager, that helps users manage their information in an efficient manner. It is also considered as a stand alone application that works with organizational server management systems such as Microsoft Exchange Server & SharePoint Server. It manages personal data which includes Mailboxes, calendar data, journal, notes, etc., and maintains a data file which collectively stores the same. Outlook stores its data in PST or Personal Storage Table File format which maintains a systematic structure of storing the information in the repository. The PST File follows a File Structure, Logical Architecture which is explained in the following research paper.

OUTLOOK DATA FILE STRUCTURE VIEW

The PST is a self-contained, stand-alone, binary structured file format that does not desire any extrinsic dependencies. Each Outlook PST file symbolizes a message store, that holds an arbitrary hierarchy of Folder objects, consisting of Message objects and Attachment objects. Report about Folder objects, Message objects, and Attachment objects are saved in the properties, which altogether consists of information about an individual item.

PST FILE LOGICAL ARCHITECTURE

The Outlook PST file structure is logically organized in three layers; Messaging layer, LTP (Lists, Tables, and Properties) layer and NDB (Node Database) layer. The diagram given below illustrates the logical hierarchy of layers, and what task is performed by each layer.



Messaging Layer

Messaging layer consists of business logic and high-level rules that permit structure of NDB and LTP layers to be interpreted and combined as Folder objects, Message objects, Attachment objects, and properties. It also defines the rules and requirements that obliges to be pursued while modifying the contents of PST file, so that the modified Outlook PST file can be read by execution of this file format.

Lists, Tables, and Properties (LTP) Layer

LTP (Lists, Tables and Properties) Layer implements high level concepts on top of NDB construct. Core elements of LTP Layer are; Property Context and Table Context, where:

Property Context: Shows Collection of Properties.

And Table Context:

Shows two dimensional table consisting of rows and columns.

- **Rows:** Shows a collection of properties.
- **Columns:** Shows which properties are within rows.

From a standpoint of high level implementation, each PC or TC (Property Context Or Table Context) is stored as a data in a single node. The LTP layer uses NID to analyze PCs and TCs, to implement them efficiently. LTP layer employs two types of data structures on top of each NDB node.

Node: It is a basic unit used in computer science. Nodes are basically an individual part of a large data structure such as; tree data structures. The Nodes contain data and may also link to other nodes.

HEAP-ON-NODE (HN)

Heap on Node is a bundle of data structure that is executed on top of the node. The Heap on node enables to sub-divide the data flow of node into small, variable sized fragments. The primary example of the Heap on Node usage is, to save various string values into a single block. On top of the Heap on Node, more complex data structures are built.

NOTE: String is a data type that is often implemented as an array of bytes or words that stores typical character and element sequence using character encoding.

BTREE-ON-HEAP (BTH)

BTree-on-Heap data structure is executed inside an HN structure, as HN structure provides a rapid way to access BTree structures. Whereas, BTree-on-Heap provides an appropriate way to search through data.

NODE DATABASE (NDB) LAYER

Node Database Layer consists of a database of nodes, which represent low level storage facilities of Outlook PST file format. From an execution point of view, an NDB layer consists of file allocation information, header, nodes, blocks, and two BTree's: Block BTree (BBT) and Node BTree (NBT).

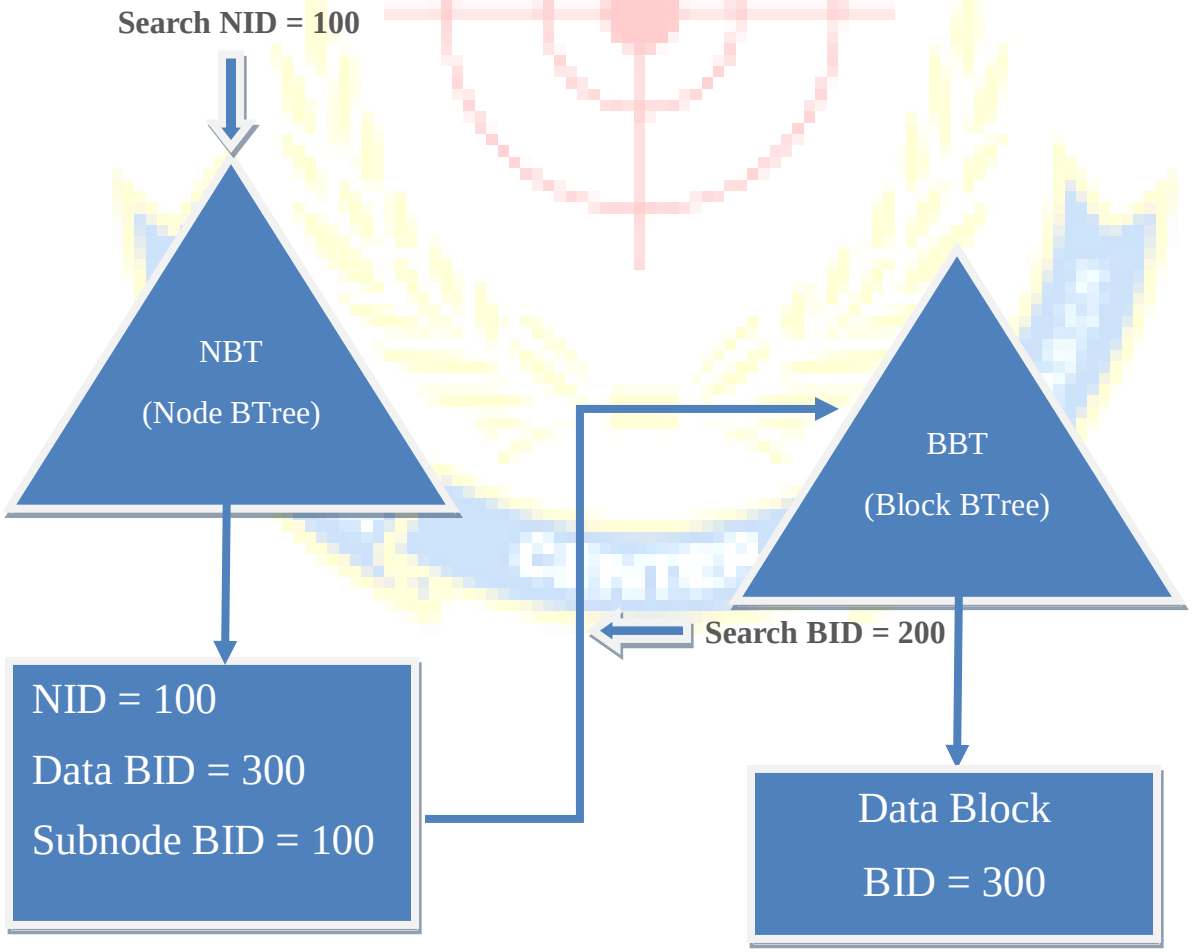
Node BTree (NBT): NBT consists of references to all of accessible nodes in PST file. Its BTree execution allows for effective searches to locate a specific node. Each node reference is represented as a set of properties that includes; NID, parent NID, data BID, and sub-node BID. Data BID points to block that consists of data associated with node and subnode BID points to block consisting of references to subnodes of this node. Top level NIDs are unique across PST and are searchable from an NBT. Subnode NID are only unique within a node and are not searchable from NBT. Parent NID is inflation for higher layer and has no sense for NDB Layer.

Block BTree (BBT): Block BTree contains references of PST file data blocks and allows execution of efficient search to detect any specific block. A reference of block is represented by four properties, which includes; its BID, CB, IB and CREF. IB is the offset within a file where the block is located, CB is the count of bytes stored within a block, CREF is the count of reference to the data stored within a block.

NDB Layer Fundamental Concepts

- Divide Outlook PST file into logical streams.
- Establish hierarchical relationships between streams.
- Provide transaction functionality when modifying data within streams.

The roots of NBT and BBT can be accessed from the PST file header.



The above figure shows how the data of a node with NID = 100 can be accessed. NBT is searchable to find records with NID = 100. Once found, the record contains BID = 200 of block that consist of a node's data. With BID, the BBT can be searched to locate block that consists of a node's data. It is always mandatory to search for both; NBT and BBT, to locate the data for a top-level node.

PST File Physical Organization

Before understanding the physical organization of a PST file, you should be familiar with the following terms:

Header: Header is located at the beginning of an Outlook PST file and contains metadata as well as root information to connect with the data structure of NDB Layer.

Root: A structure in Header that consists of the current file state.

BREF: It is a record that maps a BID to its absolute file offset location.

BID: Every block allocated to a PST file, is analyzed by using BID structure. This structure differs in size according to the file format. In case of ANSI PST file the structure is a 32 bit unsigned value, while in Unicode file, it is 64 bit unsigned.

NID: Nodes provide primary abstraction used to remark the data stored in a PST file that is not disturbed by NDB layer. Each node is identified by its NID as each NID is unique within the namespace in which it is used. Each node referenced by NBT has a unique NID, however, two subnodes of different nodes can have identical NIDs. But two subnodes of the same node have different NID.

IB: IB (Byte Index) is used to serve as an absolute offset within an Outlook PST file with respect to the file beginning. IB is an unsigned integer value and is 32 bits in ANSI versions of PST file and 64 bits in Unicode versions.

BTree: It is a Generic BTree structure widely used throughout the Outlook PST file format. In NDB Layer BTrees are the building blocks for NBT and BBT which are used to navigate and search nodes and blocks. To know about NBT and BBT structure, refer the Logical structure of a PST file.

Page: It is a fixed 512 byte structure that is used in NDB layer to represent the metadata allocation and BTree data structures.

Page Trailer: It is a 16 byte structure present at the end of the page that contains meta-data information about the page.

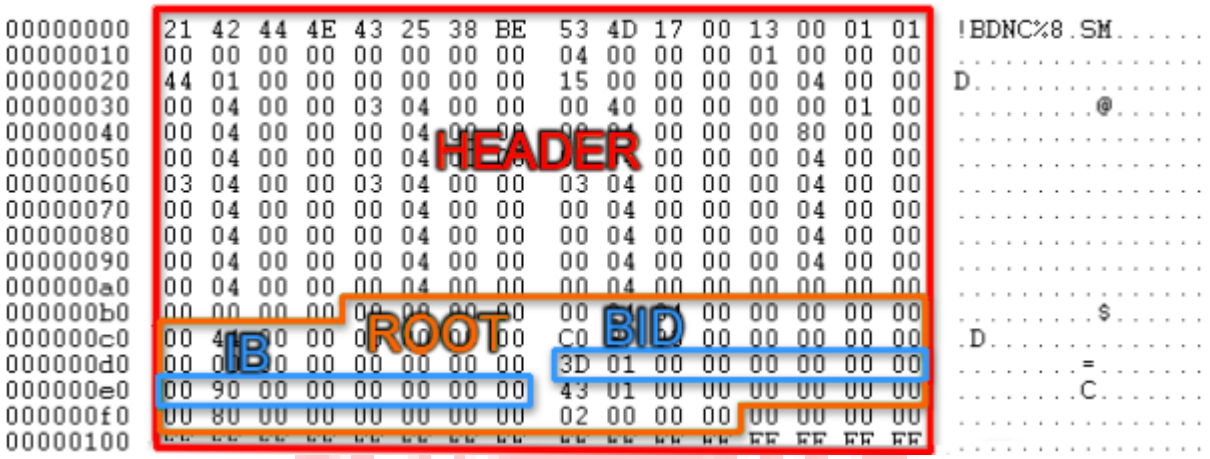
BTPage: A 512 byte page that is a part of the Node or Block BTree.

BTentry: It contains a key value of NID or BID and a reference to child BTPage in BTree.

NBTentry: It contains information about nodes and is found in BTPages with cLevel equal to 0, with ptype of ptypeNBT these are the leaf entries of NBT.

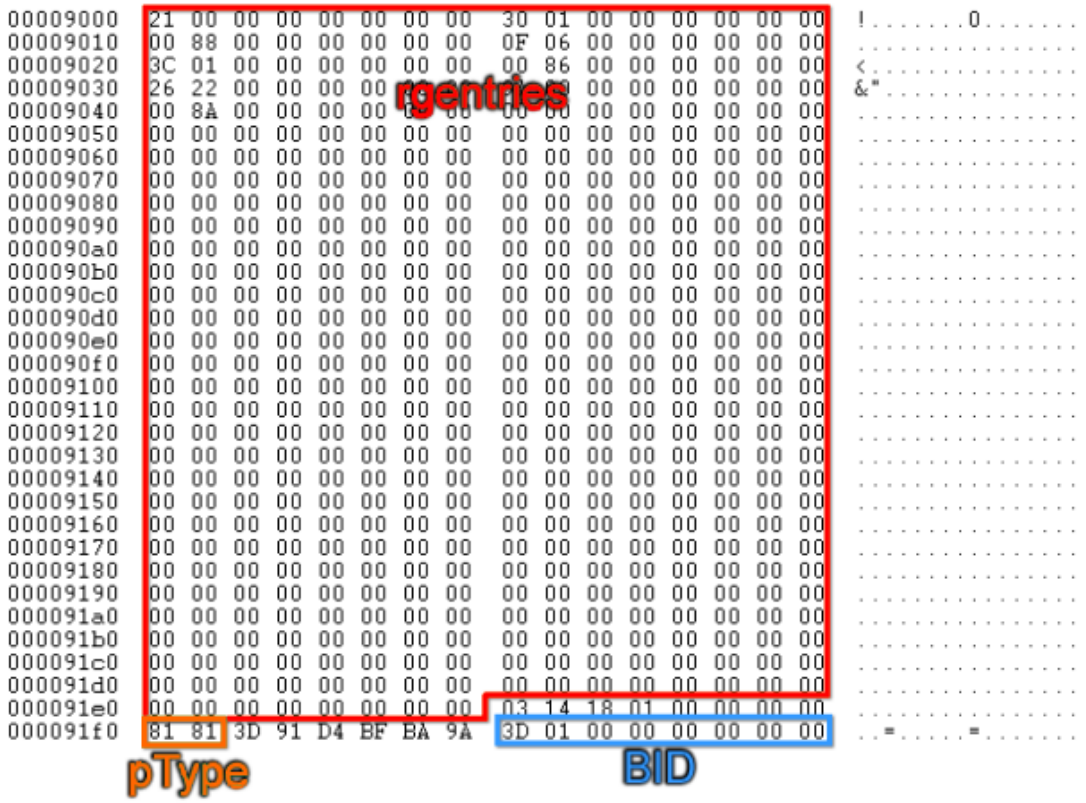
ROOT STRUCTURE

Root structure in Header consists of BREFNST structure which contains BID and IB. The IB is the second 64 bit integer value that serves an unsigned 64 bit integer and the absolute offset into the file where the first NBT page exists. In Unicode PST file, we can find IB at offset 0xE0.

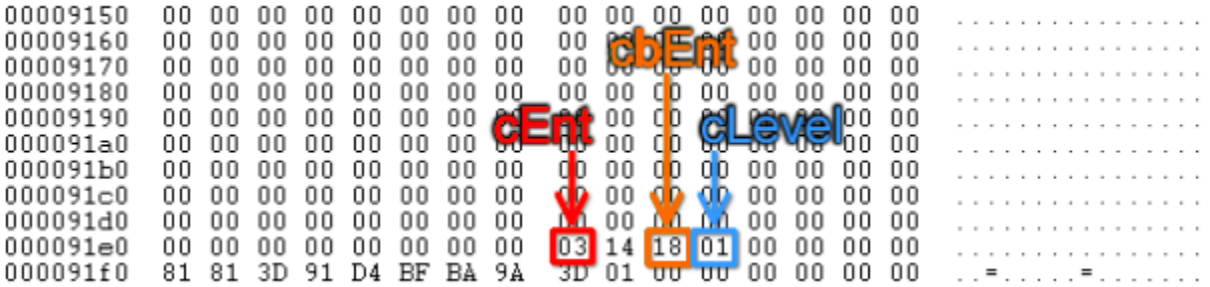


Remember, that the value is stored on disk in Endian format, so what you see in file laid out as “00 90 00 00 00 00 00 00” becomes “00 00 00 00 00 00 90 00” when read into memory. Using this, we can find out that the NBT first page is located at 0x9000 offset.

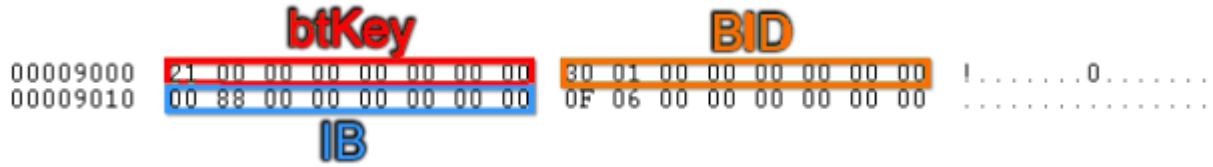
NBT Root Page



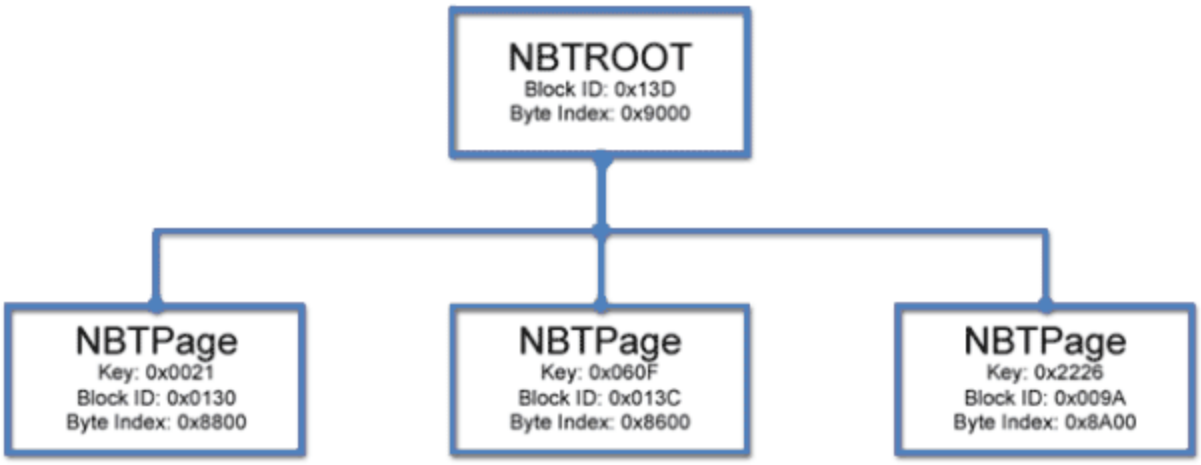
This is how the Root NBT page for a new PST file looks like with the first 488 bytes in an array of rgentries. By referring the given below table, we can verify that the pType property 0x81 is pTypeNBT. We can also match the BID of Page Trailer with BREFNBT structure in ROOT. It is best to verify that the data we are looking at, is the data structure we expect it to be.



Looking at the section right before the Page Trailer, there are three important properties that will decide how to read data in rgentries array. The number of entries in rgentries array is determined by cEnt byte, while the size of each entry is distinguished by cbEnt. The entries will be of BTentry type if cLevel byte is greater than 0. We can see that rgentries array contains 3 entries, where each one is 0x18 (24) bytes in length, and are going to be BTentry structures.



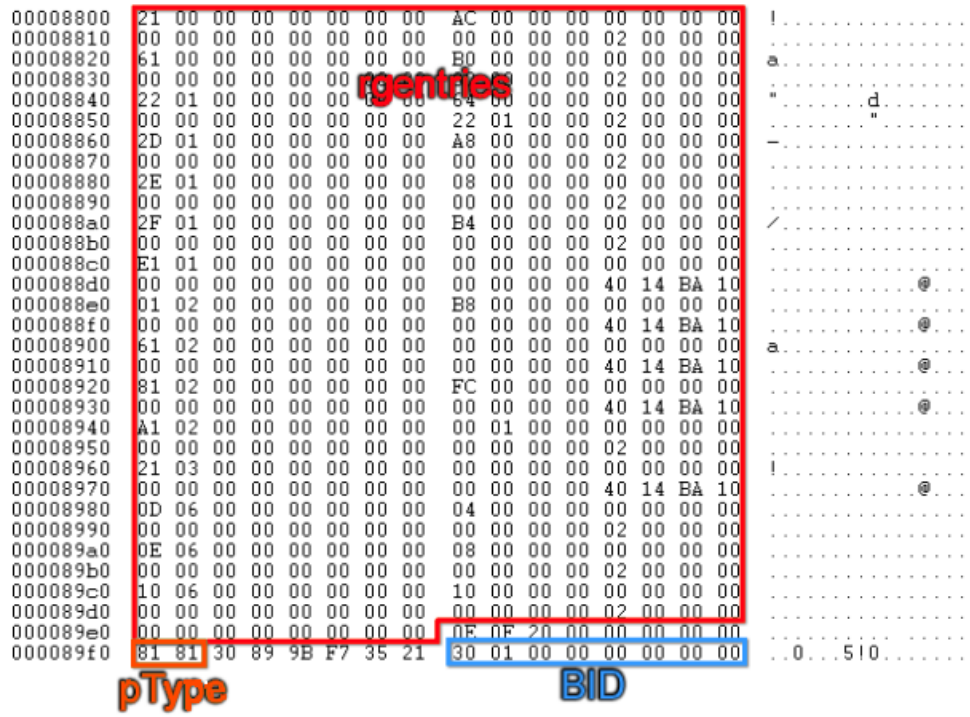
The 8 first bytes are btKey and 16 bytes that follow is BREF structure. The most important part of a BREF is IB, which is the last 8 bytes that shows where the next page in file lies. So, this entry has a key of 0x21 (33), Block ID is 0x130 and Page that it points to is at offset 0x8800.



After reading content of 3 records in rgentries array, Nodes BTree looks as:

TRAVERSING NODE BTREE

The NBT Page that Root page points to, at offset 0x8800:

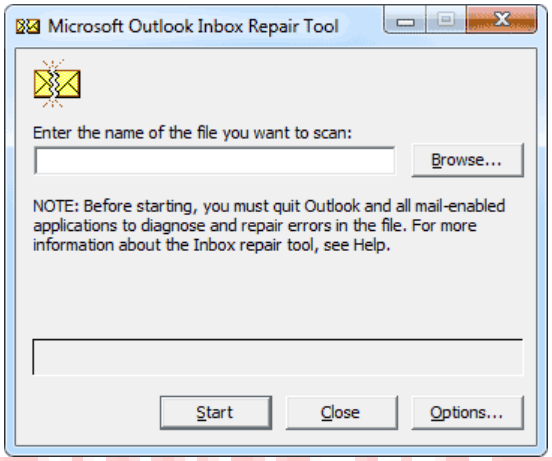


This one has recognizable page type of 0x81 (pTypeNBT) and we can authenticate that BID matches 0x130. Following the same logic as before, we see that this page contains 0x0E (14) entries, and that each one is 0x20 (32) bytes in length. cLevel is 0 this time, which means entries are NBTentry types. This shows that we are in the leaf node of the tree. If cLevel was greater than 0 we would follow the same logic as we did, with entries on NBTRoot page.

The first 8 bytes are NID which is 0x21 and next 8 bytes is BID of data block for this node (0xAC). That shows us the node ID of block is in Block Btree where the actual data is for this node. The next 8 bytes are BID of subnode block for this node. In this case, there isn't one, the next 4 bytes is parent NID. The parent NID contains the ID of the folder object that data that belong to this node belong to. In other words, if data of this node point represents is an email, the parent NID is the ID of the folder that email resides in. The last 4 bytes should be ignored as it is for alignment purpose.

IF THE PST FILE GETS MANIPULATED OR DAMAGED

Microsoft Outlook provides a free inbuilt utility Inbox repair Tool that corrects errors in internal data structures of a .pst file. It basically checks and repairs reference count and BTrees structures. An Inbox repair tool (scanpst.exe) does not have knowledge about upper-level structures, such as messages, calendars and others, built upon them.



If ScanPST determines that a specific block of structuring is unreadable or corrupted, Scanpst removes it. If that block was a part of a specific item in MS Outlook, the item will be removed during validation. This type of behavior is not expected by anyone, but the removal of an item is applicable in such circumstances. This kind of situation is rare and it will enter in ScanPST log file.

At a higher level, more visible changes that you see involves; folders and messages.

FOLDERS

Inbox Repair Tool examines every folder of a PST file and makes sure that there are correct tables associated with each folder. ScanPST checks every row in each table and makes sure that message or sub-folder exists in the system. If ScanPST does not find the message or sub-folder, then it removes rows from the table. If ScanPST does find the message or sub-folder, ScanpPST validates the message or folder. If the validation fails, then the message is considered as corrupted and is removed from the database. If the validation succeeds then, ScanPST does another examination to ensure that the recovered message values are now logical with the values in the table. Corrupted folders are Regenerated from scratch, it is unavoidable, but contains no user data.

MESSAGES

ScanPST does a basic validation of the recipient tables and attachment tables. This operation features, how a folder works with a message in it. After validation of the recipient table to guarantee recipients that are formatted correctly. ScanPST makes changes that are desired to sync valid recipient table content to recipient properties of the message. ScanPST also assures that the parent folder of the message refers to a valid folder.

No validation is clearly done on body related or subject related properties, except for the low level validation discussed earlier. The recipient replays properties that consistently change with the recovered recipient table. As this operation completes, other algorithms run to collect the orphaned messages to put them in an Orphan's folder.

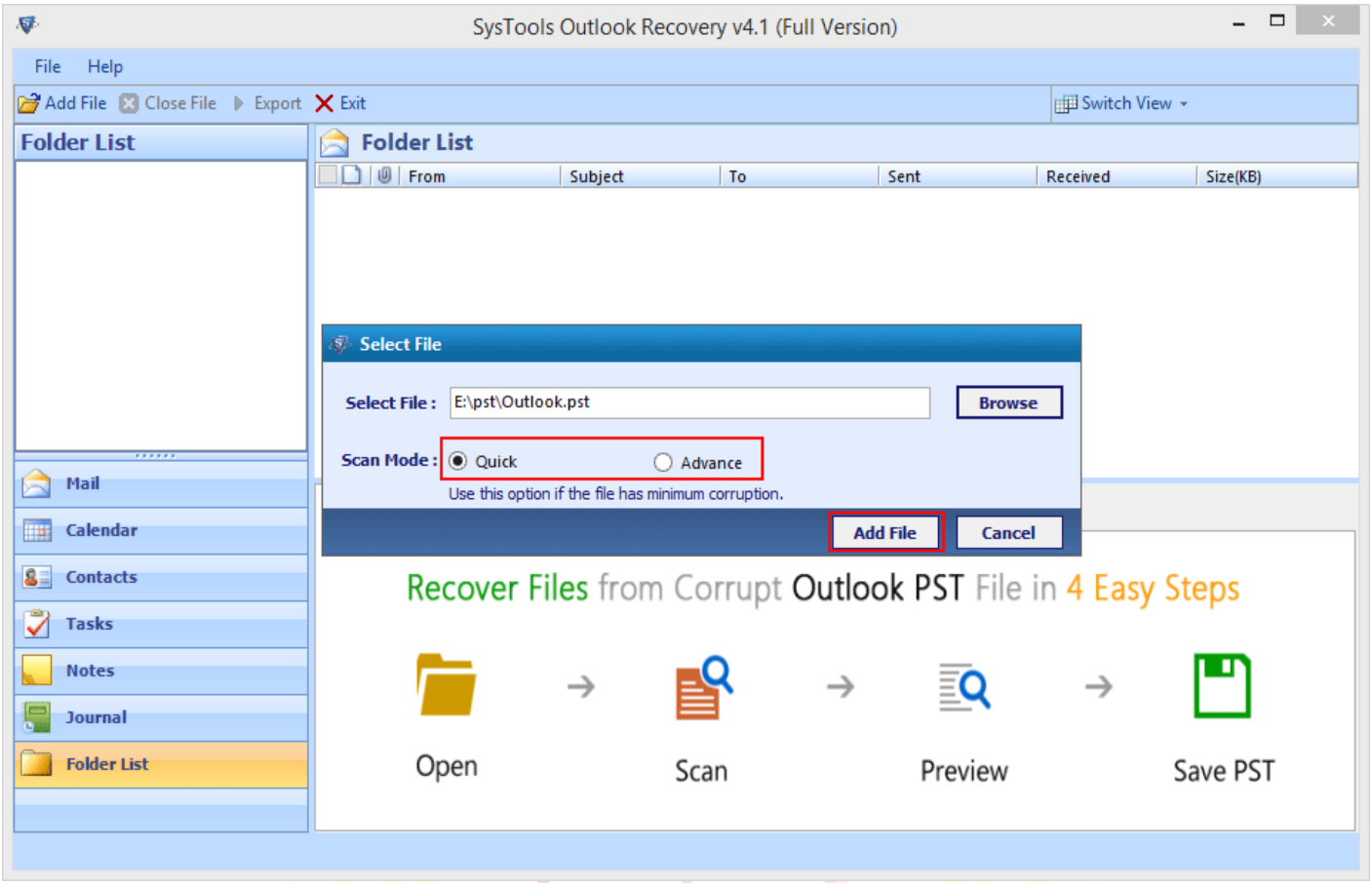
Files Created During the Repair Process

During the Repair process, ScanPST creates a file called .bak which is the copy of an original .pst file with a different extension. The .bak file is located in the same folder as original .pst file, it helps in recovering the items of PST files, that have not been recovered by the Inbox repair tool. To do so you have to import a bak.pst file and then move any additional repaired items to a newly created .pst file.

A Copy of the log file is also written to the same location as that of the PST file. This log file contains information about the steps being performed, but that's about it. The bad items are those which are no longer available in the database after recovery.

If ScanPST Fails to Repair Data

As you have seen above, ScanPST does not work at the time of high level corruption and it even deletes the items if not found in the index table. In such circumstances,usage of third party utilities is highly recommended. Software such as SysTools Outlook Recovery ,works as a dependable alternative as it recovers corrupted and encrypted PST files as well as deals with the restoration of permanently Deleted items.



CONCLUSION:

This Research Paper Sharply elaborates the complete structure of an Outlook PST file based on the multiple layers it consist of. However, even the slightest of modification when done with the internal structure of the file can lead to major consequences as discussed in the paper, to tackle which, Microsoft came up with ScanPST Tool. Regrettably. The inbuilt program developed particularly for rectifying minor damages, failed drastically at resolving file structure modification. And despite of the fact, the research paper suggests an answer to crack such cases without expecting a failure as the implementation of third party SysTools Outlook Recovery Solution.

SOURCES

http://download.microsoft.com/download/2/4/8/24862317-78F0-4C4B-B355-C7B2C1D997DB/[MS-PST].pdf

http://blogs.msdn.com/b/openspecification/archive/2010/11/30/ms-pst-how-to-navigate-the-node-btree.aspx

http://support2.microsoft.com/kb/287497

http://support.microsoft.com/kb/272227

http://www.systoolsgroup.com/outlook-recovery.html