Registry hive basics part 1

Before we get into the weeds, it will be helpful to talk about the basic building blocks of a Registry hive.

High level structures

There are 2 high level structures used in hives: the header and hbin cells.

Registry header

The Registry header is 4096 (0x1000) bytes long and contains several important items of information:

- Signature
- Last write timestamp
- Major and minor version numbers
- Root cell offset
- Length
- File name

This is of course not an exhaustive list but these are the bits of info that are most relevant.

The image below is a hive opened in WinHex at offset 0x00, or the beginning of the file:

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	ANSI ASCII
00000000	72	65	67	66	E2	5D	02	00	E2	5D	02	00	41	00	BB	1A	3B	9F	CE	01	01	00	00	00	03	00	00	00	00	00	00	00	regfâ] â] A » ;ŸÎ
00000020	01	00	00	00	20	00	00	00	00	F0	86	00	01	00	00	00	5C	00	3F	00	3F	00	5C	00	43	00	3A	00	5C	00	55	00	ð† \ ? ? \ C : \ U
00000040	73	00	65	00	72	00	73	00	5C	00	65	00	72	00	69	00	63	00	5C	00	6E	00	74	00	75	00	73	00	65	00	72	00	s e r s \ e r i c \ n t u s e r
00000060	2E	00	64	00	61	00	74	00	00	00	00	00	00	00	00	00	0A	AD	7A	13	7E	AE	E3	11	80	BB	90	B1	1C	1C	CB	90	. d a t -z ~®ã €» ± Ë
00000080	0A	AD	7A	13	7E	AE	E3	11	80	BB	90	B1	1C	1C	CB	90	00	00	00	00	0B	AD	7A	13	7E	AE	E3	11	80	BB	90	B1	-z ~®ã €» ± Ë -z ~®ã €» ±
000000A0	1C	1C	CB	90	72	6D	74	6D	A5	30	23	A1	15	25	D0	01	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	Ë rmtm¥0#¡ %Ð
000000C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000000E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	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	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000001A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000001C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000001E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	D6	FE	93		Öþ"

[http://4.bp.blogspot.com/-1KFZuJvH4hw/VK1NZCEPF4I/AAAAAAAAAIk/nCYGl6l3s6w/s1600/Reg2.png]

Signature

The signature is found at offset 0x0 and is 4 bytes long. It is the ASCII string 'regf' and all hive files will start with this signature.

Last write timestamp

The last write timestamp is found at offset 0xc and is stored as a 64-bit integer that represents the number of 100-nanosecond intervals since January 1, 1601 (UTC). This is a Windows FILETIME.

In the screenshot above, the timestamp is 0x01CE9F3B1ABB0041. You will notice this is "backwards" to how it appears in the screenshot. This is because it is stored in little endian format. The same number represented in decimal is 130216515440672833.

Using a tool like DCode we can get the actual date by entering in the hex values and selecting the appropriate

endianness. Entering in the hex value above and choosing Big endian yields 'Thu, 22 August 2013 13:25:44 UTC' as our timestamp.

In .net, there is a function to convert a FILETIME to a DateTimeOffset (DateTimeOffset.FromFileTime strangely enough).

In my Registry parser, all timestamps are converted to DateTimeOffsets, which allows for easily converting to any timezone. The default is UTC.

Major and minor version numbers

The major version lives at offset 0x14 and is stored as a 32-bit integer. The minor version lives at offset 0x18 and is stored the same way. In the screenshot above you will see these values are also stored in little endian format (as are just about all numerical values).

The version numbers are important because certain behavior is only found in certain version numbers. For example, in version 1.4 and greater, large values are stored differently than in versions less than 1.4. We will see more on this when we discuss vk cell records which are used to store values.

Root cell offset

The root cell offset lives at 0x24 and points to the top level key of a hive. It is stored as a 32-bit unsigned integer.

Unsigned integers are used throughout the Registry so it is important to understand the difference between a signed and unsigned integer. An unsigned integer can only be positive. The maximum value for a 32-bit unsigned integer is 4,294,967,295. This means an unsigned integer can be between 0 and 4,294,967,295.

Stepping back again, since we are talking about a 32-bit number, we can divide 32 by 8 to get to the number of bytes, 4.

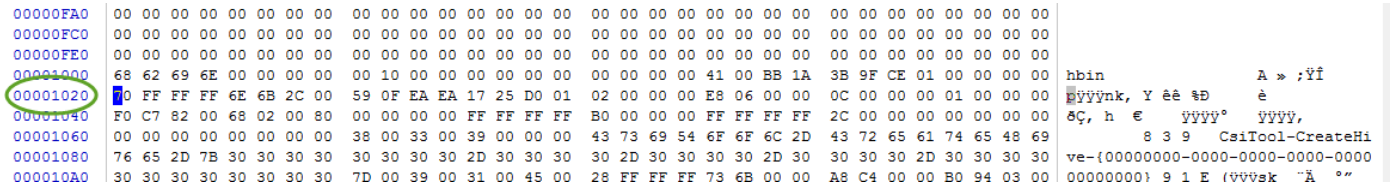
In hex, the maximum value for an unsigned 32-bit integer is 0xFFFFFFFF which is 8 F's. This is 4 bytes long (FF FF FF FF) and 4 bytes * 8 bits gets us back to 32 bits.

For signed integers, think of sliding the left side of the range into the negative numbers. When you reach the "middle" you now have your maximum and minimum for a signed integer. In the case of a 32-bit signed integer, the minimum is –2,147,483,648 and the maximum is 2,147,483,647. As you would expect, adding the absolute value of these two numbers together equals 4,294,967,295.

When we get into the different cell and list records (specifically the size), negative vs. positive numbers will be important.

Getting back to the root cell offset, in the screenshot above, the root offset is 0x20. This is the **relative** position of the root cell. All offsets in the Registry are relative to the first hbin record (which we will talk about after we finish with the Registry header).

Recall a hive's header is 4096 (0x1000) bytes long. This means the first hbin cell will be found at absolute offset 0x1000. To calculate the absolute position of the root cell, we have to add 0x1000 to the relative position of the root cell, 0x20. This results in an offset of 0x1020 and if we look at that offset, we can see the data structure for the root cell.



[http://3.bp.blogspot.com/-slFDpo6Tu7s/VK1mE1ZUr2I/AAAAAAAAAI0/Rbj3OgIKvsI/s1600/1020.png]

As you can see in the screenshot above, there is an hbin header at 0x1000 and our root cell lives inside that hbin cell. We will discuss hbin cells after finishing up the header.

Length

The length of the registry hive is found at offset 0x28. It is stored as a 32-bit unsigned integer and represents the total size of all currently in-use hbin cells in the registry hive. In the first screenshot we see the length is 0x0086F000, or 8,843,264, bytes.

Note that this is NOT equal to the size of the registry hive on disk. The registry header size is not included in this value, nor is any extra data at the end of the registry hive.

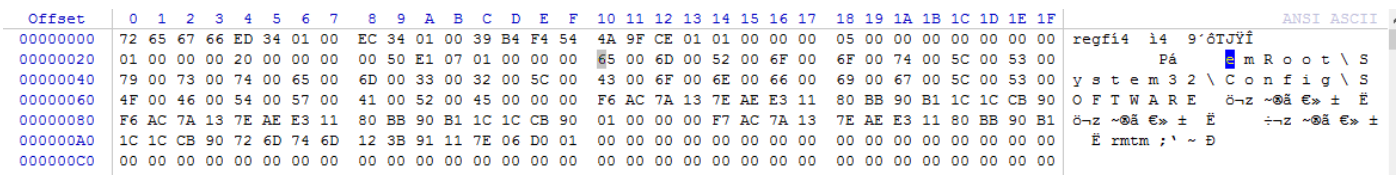
Length can be used to ensure that all in-use areas of a registry hive have been read by a parser. My registry parser uses this value as a way to check that the number of bytes read when parsing equals the number of bytes stored in the Length field. In some cases there is erroneous data or zeros beyond Length. My parser will check any data beyond Length and if its all zeros, its ignored. If any non-zero data is found, it is reported as a warning.

File name

Finally, the file name is embedded in the registry. it can be found at offset 0x30. It is stored in UTF-16 little-endian format whose maximum length is 64. The string is terminated by a NUL (end of string) character. In the first screenshot above, we can see the file name is "\??\C:\Users\eric\ntuser.dat" and from this, we can determine the user profile this hive came from.

Note: Joachim Metz's research has shown that not all hives have a file name embedded in them. When the name is present, it will be reported.

Other registry hives do not contain the absolute path, but by looking at the rightmost portion of the filename, we can determine what kind of hive file we are looking at. As an example, consider the following screenshot:



[http://3.bp.blogspot.com/-9JIT0ZI2-vQ/VK1rP668CUI/AAAAAAAAAJE/aVumHfmwQy8/s1600/filename.png]

This is a SOFTWARE hive that lives in System32\Config.

hbin cells

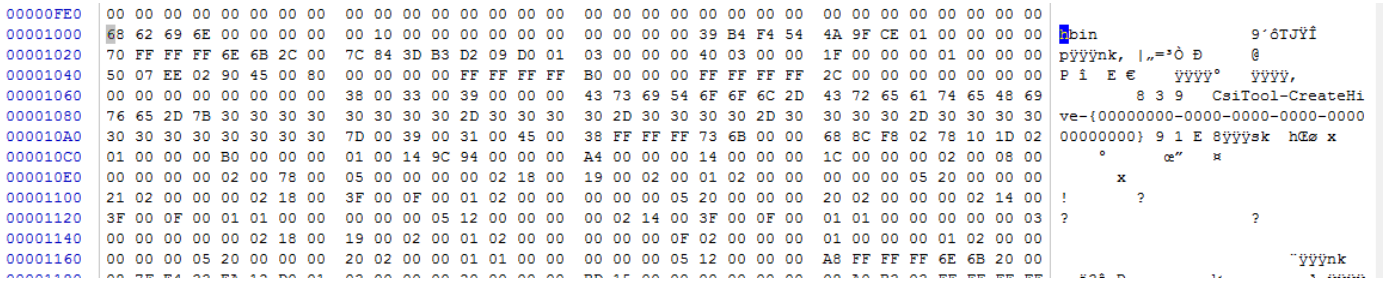
hbin cells are the "containers" for all other records found in registry hives.

hbin cells are multiples of 4096 (0x1000) bytes long and contains several important items of information:

- Signature
- FileOffset
- Size

This is of course not an exhaustive list but these are the bits of info that are most relevant.

The image below is a hive opened in WinHex at offset 0x1000, the first hbin cell:



[http://1.bp.blogspot.com/-j0oLvWdihv4/VK1uLhOOk9I/AAAAAAAAAJQ/o1Y4ONQLGFs/s1600/hbin.png]

Signature

The signature is found at offset 0x0 of the hbin cell and is 4 bytes long. It is the ASCII string 'hbin' and all hbin cells will start with this signature.

FileOffset

FileOffset is the **relative** offset of this hbin ***in relation to all other hbin cells***. It is found at relative offset 0x4 in the hbin cell and is stored as an unsigned 32-bit integer.

In the above screenshot, we can see the offset is 0x00000000. Since this is the first hbin cell, it makes sense the relative offset would be 0. To calculate the absolute offset of an hbin cell, we simply add 0x1000 to FileOffset.

To get the absolute offset to FileOffset, add 0x1000 (the registry header), the FileOffset value (the hbin's relative offset), and 0x4, which in this example would be 0x1004 (0x1000 + 0x00 + 0x04).

Note: It is critical to understand this point if you plan on validating the findings of parsers with a hex editor. Since the registry stores offsets as relative offsets, you will not find the data structures if you go to the offset as stored in various registry records in a hex editor. You must always add 0x1000 to the offset to find the data structure in a hex editor!

As we will see in a future post, this relative offset is the basis for the offsets to other data structures that we will pull out of hbin cells.

The good news is my registry parser stores both the relative offset and the absolute offset to any given data structure to make finding data structures in a hex editor easier.

Size

The size is found at relative offset 0x8 in the hbin cell. In the example above, the value is 0x1000 (again, its stored in little endian format). 0x1000 in decimal is 4096. Most hbin cells are 0x1000 bytes long, but this is not always the case. However, in every case the Size will be a multiple of 4096.

Now that you have an understanding of the basic building blocks of the registry we can start diving into the more interesting bits, including records that represent registry keys (and subkeys), values, and the lists that tie the two together.

Next up, the NK record [http://binaryforay.blogspot.com/2015/01/registry-hive-basics-part-2-nk-records.html] !

Posted 8th January by ERZ

Labels: Registry

1 View comments

davnads January 8, 2015 at 9:34 AM

Great write up. Thanks!!

Reply

davnads January 8, 2015 at 9:34 AM

Great write up. Thanks!!

Reply

Enter your comment...

Comment as: ggyy (Google)

Sign out

Publish Preview Notify me