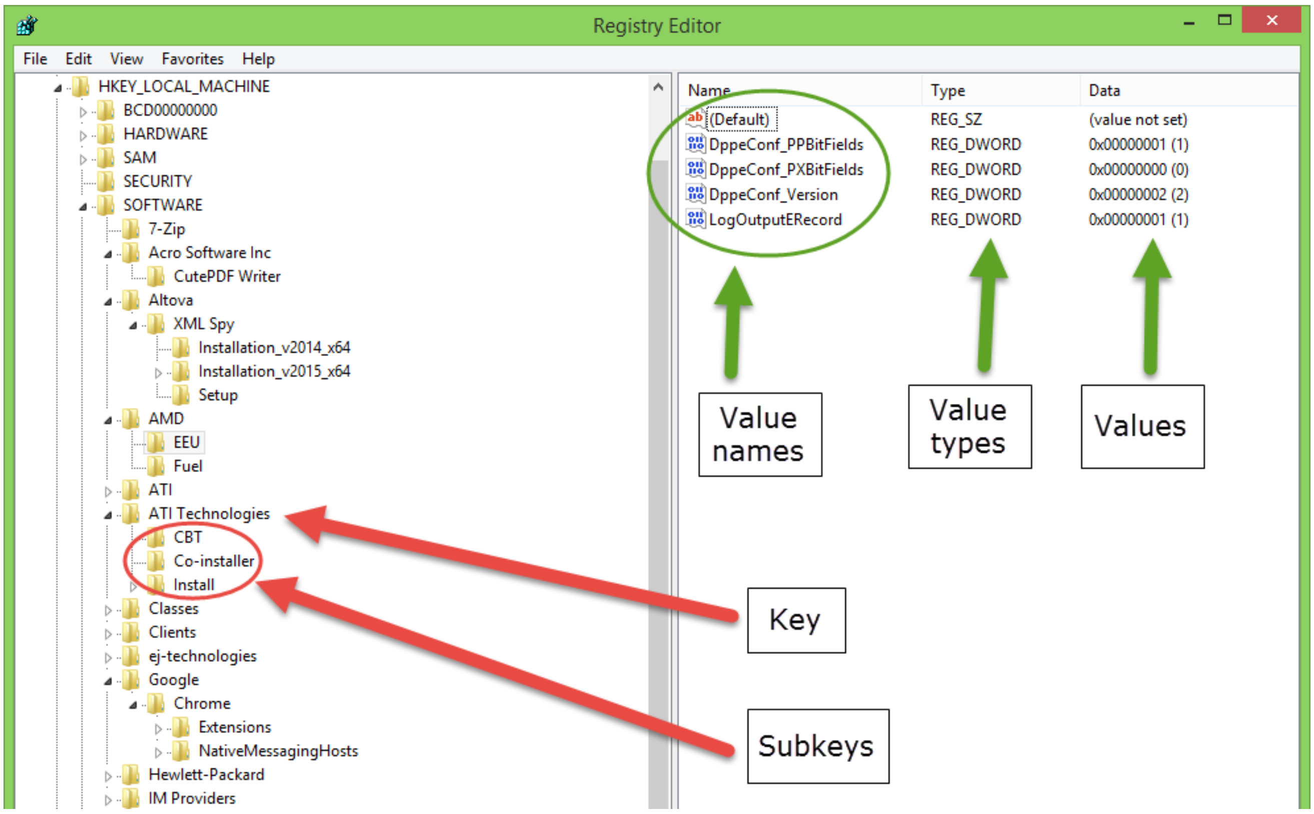
Registry hive basics part 2: NK records

Background

Most people are familiar with viewing the Registry in a program like regedit.exe. regedit.exe ships with all versions of Windows and allows a user to view the basic components of the Registry. The screen shot below illustrates the different concepts that make up a hive (or at least those exposed by regedit.exe)



```
HKEY_LOCAL_MACHINE
  BCD00000000
  HARDWARE
  SAM
  SECURITY
  SOFTWARE
    7-Zip
    Acro Software Inc
      CutePDF Writer
    Altova
      XML Spy
        Installation_v2014_x64
        Installation_v2015_x64
        Setup
    AMD
      EEU
      Fuel
    ATI
    ATI Technologies
      CBT
      Co-installer
      Install
    Classes
    Clients
    ej-technologies
    Google
      Chrome
        Extensions
        NativeMessagingHosts
    Hewlett-Packard
    IM Providers
```

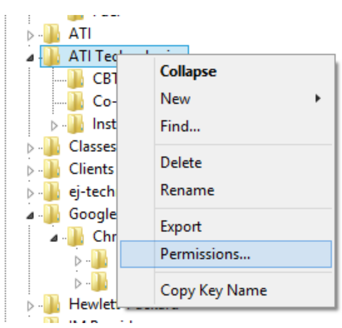
Name	Type	Data
(Default)	REG_SZ	(value not set)
DppeConf_PPBitFields	REG_DWORD	0x00000001 (1)
DppeConf_PXBitFields	REG_DWORD	0x00000000 (0)
DppeConf_Version	REG_DWORD	0x00000002 (2)
LogOutputERecord	REG_DWORD	0x00000001 (1)

[http://3.bp.blogspot.com/-BXj6dmqxwKA/VMae_q5cFwI/AAAAAAAAAKE/T7EWwIqVLNM/s1600/regedit.png]

On the left we have a tree comprised of keys and subkeys that are related to each other hierarchically. A key becomes a subkey when it is a child of a given key. In the example above, *CBT* is a subkey of the *ATI Technologies* key. There is no difference between a key and a subkey's structure.

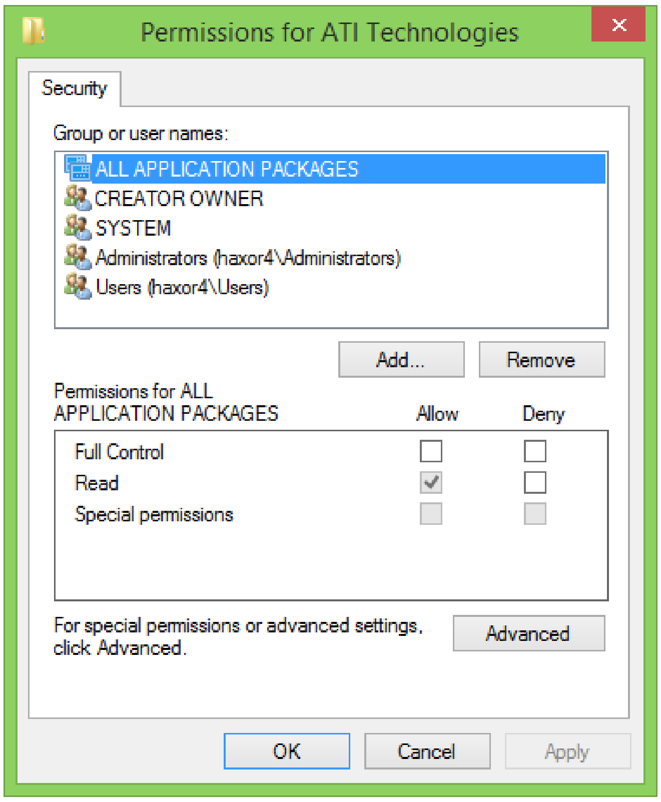
After selecting a key on the left, any values contained in that key will be displayed in the right pane. regedit makes available three pieces of information when it comes to values: the value name (Name column), the value type (Type column), and the value itself (Data column).

regedit also allows for viewing permissions on different keys in the Registry. To access these permissions, right click on a key and select Permissions...



[http://1.bp.blogspot.com/-fgBlGpbn1-I/VMag6zYV8YI/AAAAAAAAAKU/bkmnGKV5JGM/s1600/accessPerms.png]

As you would expect. clicking on Permissions shows the current permissions for the selected registry key.


Permissions for ATI Technologies

Security

Group or user names:

- ALL APPLICATION PACKAGES
- CREATOR OWNER
- SYSTEM
- Administrators (haxor4\Administrators)
- Users (haxor4\Users)

Permissions for ALL APPLICATION PACKAGES

	Allow	Deny
Full Control	☐	☐
Read	✓	☐
Special permissions	☐	☐

For special permissions or advanced settings, click Advanced.

[http://3.bp.blogspot.com/-i0Y5gznuWw8/VMag62L9yuI/AAAAAAAAAKQ/CfeWyU-XJ3Y/s1600/examplePerms.png]

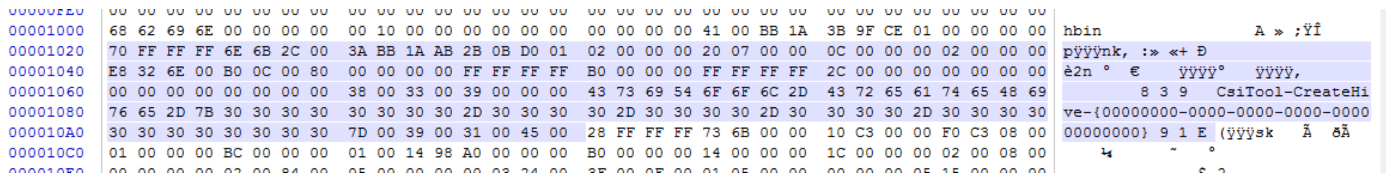
To summarize, regedit exposes three types of information: **keys**, **values**, and **permissions**.

These three types make up the bulk of the structure in a registry hive.

When looking at the structure of a Registry hive as it exists in disk however, some new terms are needed.

NK record

An NK (named key) record contains the information necessary to define a key (and subkeys as well). An example of an NK record as it exists on disk is shown below:



```
00001000  68 62 69 6E 00 00 00 00  00 10 00 00 00 00 00 00  00 00 00 00 41 00 BB 1A  3B 9F CE 01 00 00 00 00  hbin              A » ;ŸÎ
00001020  70 FF FF FF 6E 6B 2C 00  3A BB 1A AB 2B 0B D0 01  02 00 00 00 20 07 00 00  0C 00 00 00 02 00 00 00  pÿÿÿnk, :» «+ Ð
00001040  E8 32 6E 00 B0 0C 00 80  00 00 00 00 FF FF FF FF  B0 00 00 00 FF FF FF FF  2C 00 00 00 00 00 00 00  è2n °  €    ÿÿÿÿ°   ÿÿÿÿ,
00001060  00 00 00 00 00 00 00 00  38 00 33 00 39 00 00 00  43 73 69 54 6F 6F 6C 2D  43 72 65 61 74 65 48 69          8 3 9   CsiTool-CreateHi
00001080  76 65 2D 7B 30 30 30 30  30 30 30 30 2D 30 30 30  30 2D 30 30 30 30 2D 30  30 30 30 2D 30 30 30 30  ve-{00000000-0000-0000-0000-0000
000010A0  30 30 30 30 30 30 30 30  7D 00 39 00 31 00 45 00  28 FF FF FF 73 6B 00 00  10 C3 00 00 F0 C3 08 00  00000000} 9 1 E (ÿÿÿsk   Ã  ðÃ
000010C0  01 00 00 00 BC 00 00 00  01 00 14 98 A0 00 00 00  B0 00 00 00 14 00 00 00  1C 00 00 00 02 00 08 00      ¼      ˜    °
```

[http://3.bp.blogspot.com/-srZEdR_cCLs/VMaj2o5nGCI/AAAAAAAAAKk/VYzSpz4KAyI/s1600/nk.png]

In total, there are up to 25 pieces of information in an NK record. Key pieces of information available in an NK record include:

- Size
- Signature
- Flags
- Last write time
- Parent Cell Index
- Subkey count
- Subkey Cell Index
- Value count
- Value Cell Index
- Security Cell Index
- Name

Size

The size starts at offset 0x0 and is stored as a signed 32-bit integer. In the screenshot above, the hex value is 0x70FFFFFF. Like we saw earlier, this value is stored in little endian format. Converting the value to decimal results in -144 (yes, negative 144).

It may seem strange to have a record with a negative size. How can a record have a negative size? Well it really doesn't. The size of the record is 144 bytes long, but the sign denotes which records are in use and which records are free.

If a record (of any kind) has a negative size, it is in use. If a record has a positive size, it is considered free. This means that any time we see a negative size, that record should be referenced by some other record in the "active" Registry. By active I mean you should be able to find said record using regedit.exe. To put it another way, records with negative sizes aren't deleted.

Signature

The signature for an NK record is found at offset 0x04. The signature for an nk record is the ASCII string "nk" which is 0x6E6B in hex. You can verify this by using a lookup table [http://www.ascii-code.com/] .

Flags

Flags are used to denote different properties for the NK record. The flags value lives at offset 0x06 and is two bytes in length. In the example above, the flags value is 0x2C00, or simply 0x2C. Some of the most common flag values are:

- Compressed Name = 0x0020 (100000 in binary)
- Hive Entry Root Key = 0x0004 (000100 in binary)
- Hive Exit = 0x0002 (000010 in binary)
- No Delete = 0x0008 (001000 in binary)

A flag value of 0x2C equates to **CompressedName | HiveEntryRootKey | NoDelete**, but how does 0x2C become three flag values?

Looking at the list above, we can see each flag corresponds to a certain hex value and equivalent binary value. 0x2C in binary form is 101100. If we bitwise AND the flags above with our flags value, we can determine which flags are present:

101100 & 100000 == 100000 (which means the Compressed Name flag is set)
101100 & 001000 == 001000 (which means the No Delete flag is set)
101100 & 000100 == 000100 (which means the Hive Entry Root Key flag is set)
101100 & 000010 == 000000 (which means the Hive Exit flag is ***NOT*** set)

If we look at the values of each flag and add them together (100000, 001000, and 000100) we get 101100, or 0x2C. =)

You can use this website [http://www.miniwebtool.com/bitwise-calculator/] to test this yourself.

The Compressed Name flag means the Name of this nk record is stored in ASCII as opposed to Unicode (more on this later).

The Hive Entry Root Key flag is very important as it determines the root node of the Registry hive. Only one node in a hive should have this flag set. My Registry parser looks for an nk record with this flag set in addition to looking at the offset to the Root Cell Index in the Registry header.

Last write time

The last write timestamp is found at offset 0x08 and is 8 bytes long. This timestamp is the last time this nk record was updated. It is stored as a 64-bit value that represents the number of 100-nanosecond intervals that have elapsed since 12:00 midnight, January 1, 1601 A.D. (C.E.) Coordinated Universal Time (UTC).

The last write timestamp for the record above is 11/28/2014 4:52:17 PM +00:00.

These are the same types of structures we already discussed in part 1 [http://binaryforay.blogspot.com/2015/01/registry-hive-basics.html] .

Parent Cell Index

The parent cell index is found at offset 0x14 and is 4 bytes long. In the example above, the parent cell index is 0x20070000, or 0x720. In the case of an NK record that has the HiveEntryRootKey flag set, this value doesn't appear to point to an NK record since it is the root of the Registry hive.

For all other NK records, the parent cell index will point to the NK record's parent NK record. The parent cell index lets you go "backwards" or "up" from a given subkey until you get to the root node. To go "forwards" or "down" from a given subkey, we need to look at the Subkey cell index.

This is a very important field when recovering and reassociating deleted NK records because when an NK record is deleted, the Subkey count is reset to 0 and the Subkey cell index is overwritten. The Parent cell index however, is not, and by looking at the Parent cell index we can reassociate subkeys with their parent.

Subkey count

The subkey count is found at offset 0x18 and is 4 bytes long. The subkey count above is 0x0C000000, or 12 in decimal. This means this NK record will have 12 subkeys.

Subkey Cell Index

The subkey cell index is found at offset 0x20 and is 4 bytes long. The subkey cell index contains an offset to one of several list structures (to be discussed in a separate post). This list structure in turn contains offsets (or lists of offsets to more lists!) to the various subkeys. Each offset is stored as an unsigned 32-bit integer (4 bytes long). For active NK records, the list (or lists) should contain a number of offsets equal to Subkey count. With deleted NK records things can get tricky, but that is another post for another day!

In the example above, the Subkey cell index is 0xE8326E00, or **relative offset** 0x6E32E8. Recall from an earlier post we need to add 0x1000 to this value to find the actual list in a hex editor.

All of the various lists will be explained after the basic record types.

Value count

The value count is found at offset 0x28 and is 4 bytes long. This is similar to Subkey count but instead of the number of subkeys this NK record has, this is the number of values contained in the NK record.

In the example above, the value count is 0x00000000, or 0.

Value Cell Index

The value cell index is found at offset 0x2C and is 4 bytes long. The value cell index contains an offset to a data record that contains a list of offsets to VK records. VK records will be discussed in depth in part 3. A data record has very little in the way of formal structure other than a size. It does not contain a signature like other record types. Certain lists use data records to store their data (and in turn do have a signature, "db"). These will be covered in a future post.

In the example above, the value cell index is 0xFFFFFFFF. This value means this NK record has no values associated with it. You may also see 0xFFFFFFFF in other cell indexes such as Subkey cell index (a given key has no subkeys) and class cell index (a given key has no class name).

Security Cell Index

The security cell index is found at offset 0x30 and is 4 bytes long. The security cell index contains an offset to an SK record. SK records will be discussed in depth in part 4.

Name

The name of the NK record begins at offset 0x50. The length of the name is 2 bytes long starting at offset 0x4C. In the example above, the length value is 0x3900, or 57 in decimal.

Recall from above that the Compressed Name flag was set for this nk record. This means that the name is stored as ASCII and therefore must be decoded as such.

The raw value for the name looks like this:

43 73 69 54 6F 6F 6C 2D 43 72 65 61 74 65 48 69 76 65 2D 7B 30 30 30 30 30 30 30 30 2D 30 30 30 30 2D 30 30 30 30 2D 30 30 30 30 2D 30 30 30 30 30 30 30 30 30 30 30 30 7D

which, when converted to an ASCII string, reads:

CsiTool-CreateHive-{00000000-0000-0000-0000-000000000000}

This website [http://rishida.net/tools/conversion/] can be used to manually decode the hex values to strings.

Recall that all records are a multiple of 8 bytes (144/8 == 18). If we add the length of the Name to where it started, at offset 0x50, we get 0x89, or 137 bytes.

To get to 144 bytes (the next multiple of 8), padding is used. This padding may be all zeros or could be remnants of a previously existing record.

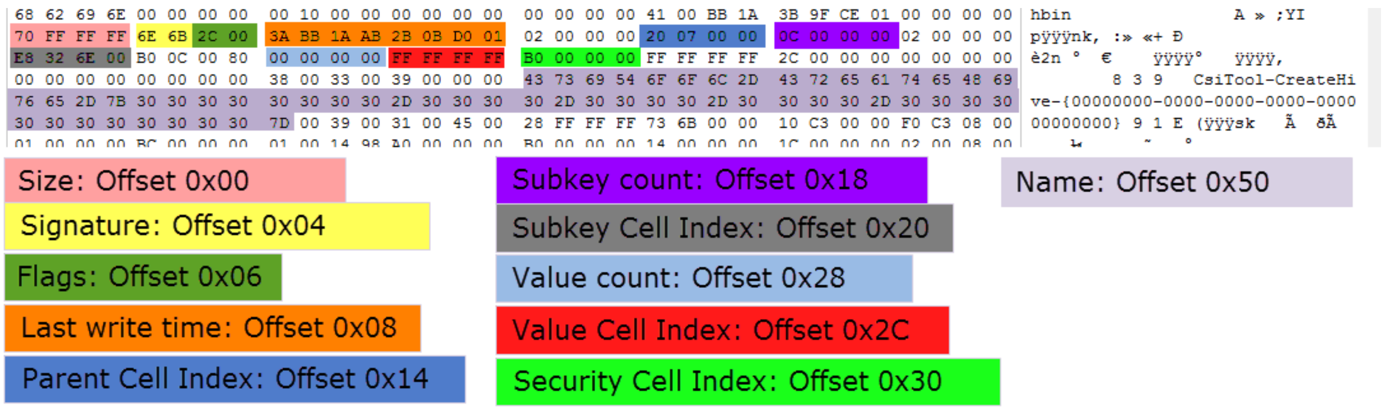
In the screenshot above, the padding values are:

00 39 00 31 00 45 00

If we then take these 7 bytes and add them to the 137 bytes from the NK record, we get 144 bytes, which is precisely what we were expecting.

Putting it all together

The most important pieces of information are highlighted below:



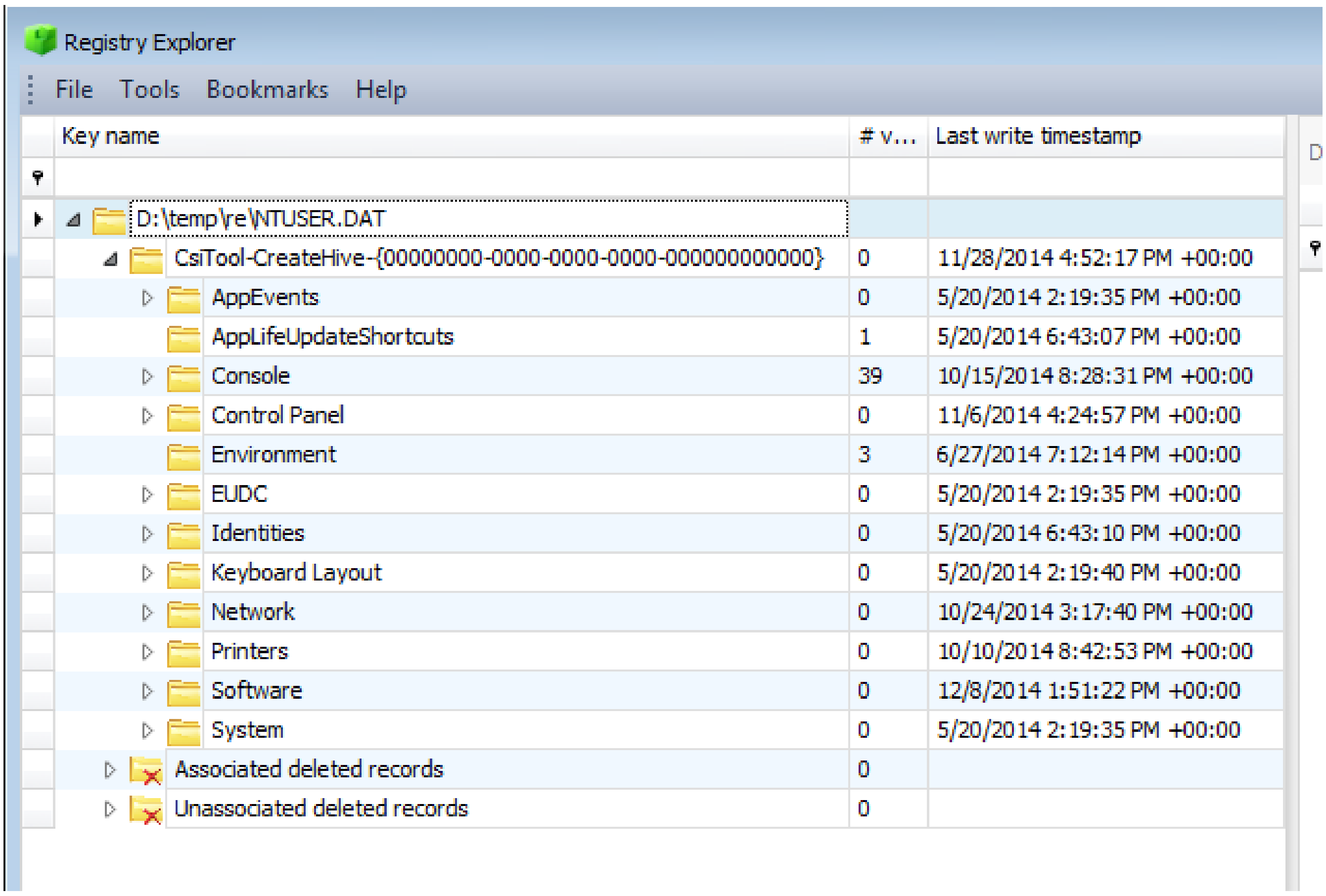
```
68 62 69 6E 00 00 00 00  00 10 00 00 00 00 00 00  00 00 00 00 41 00 BB 1A  3B 9F CE 01 00 00 00 00  hbin              A » ;YI
70 FF FF FF 6E 6B 2C 00  3A BB 1A AB 2B 0B D0 01  02 00 00 00 20 07 00 00  0C 00 00 00 02 00 00 00  pÿÿÿnk, :» «+ Ð
E8 32 6E 00 B0 0C 00 80  00 00 00 00 FF FF FF FF  B0 00 00 00 FF FF FF FF  2C 00 00 00 00 00 00 00  è2n °  €    ÿÿÿÿ°   ÿÿÿÿ,
00 00 00 00 00 00 00 00  38 00 33 00 39 00 00 00  43 73 69 54 6F 6F 6C 2D  43 72 65 61 74 65 48 69          8 3 9   CsiTool-CreateHi
76 65 2D 7B 30 30 30 30  30 30 30 30 2D 30 30 30  30 2D 30 30 30 30 2D 30  30 30 30 2D 30 30 30 30  ve-{00000000-0000-0000-0000-0000
30 30 30 30 30 30 30 30  7D 00 39 00 31 00 45 00  28 FF FF FF 73 6B 00 00  10 C3 00 00 F0 C3 08 00  00000000} 9 1 E (ÿÿÿsk   Ã  ðÃ
01 00 00 00 BC 00 00 00  01 00 14 98 A0 00 00 00  B0 00 00 00 14 00 00 00  1C 00 00 00 02 00 08 00
```

Size: Offset 0x00
Signature: Offset 0x04
Flags: Offset 0x06
Last write time: Offset 0x08
Parent Cell Index: Offset 0x14
Subkey count: Offset 0x18
Subkey Cell Index: Offset 0x20
Value count: Offset 0x28
Value Cell Index: Offset 0x2C
Security Cell Index: Offset 0x30
Name: Offset 0x50

[http://1.bp.blogspot.com/-w3XlpphV-1c/VMarxYVCqrI/AAAAAAAAAK0/XSHqZUKH-kI/s1600/nkbreakdown.png]

For a full breakdown of the NK record layout, review the documents posted earlier.

If we look at this NK record in Registry Explorer, it looks like this:



Key name	# v...	Last write timestamp
D:\temp\re\NTUSER.DAT		
CsiTool-CreateHive-{00000000-0000-0000-0000-000000000000}	0	11/28/2014 4:52:17 PM +00:00
AppEvents	0	5/20/2014 2:19:35 PM +00:00
AppLifeUpdateShortcuts	1	5/20/2014 6:43:07 PM +00:00
Console	39	10/15/2014 8:28:31 PM +00:00
Control Panel	0	11/6/2014 4:24:57 PM +00:00
Environment	3	6/27/2014 7:12:14 PM +00:00
EUDC	0	5/20/2014 2:19:35 PM +00:00
Identities	0	5/20/2014 6:43:10 PM +00:00
Keyboard Layout	0	5/20/2014 2:19:40 PM +00:00
Network	0	10/24/2014 3:17:40 PM +00:00
Printers	0	10/10/2014 8:42:53 PM +00:00
Software	0	12/8/2014 1:51:22 PM +00:00
System	0	5/20/2014 2:19:35 PM +00:00
Associated deleted records	0	
Unassociated deleted records	0	

[http://2.bp.blogspot.com/-PNqWZbZ_Vq4/VMbwX50Qa4I/AAAAAAAAALE/KLqk1uUV32I/s1600/nkInViewer.png]

Here you can see the hive name from above, the last modified date, the number of values, and the 12 subkeys.

Registry hive basics part 3 will cover the VK [http://binaryforay.blogspot.com/2015/01/registry-hive-basics-part-3-vk-records.html] , or value key record and part 4 will cover the SK [http://binaryforay.blogspot.com/2015/02/registry-hive-basics-part-4-sk-records.html] , or security key record.

Posted 27th January by ERZ

Labels: Registry

2 View comments

esc4rg0t October 23, 2015 at 9:56 AM

Hi there,...
nice information on registry you've got there! I'm currently dissecting registry format and obviously found your website. Now, regarding NK records:
I've seen that not all "hbins" begin with "nk" records. Most of the times, from the beginning of "hbins", you add 0x20 and there the "nk" record starts. But sometimes, that's just complete,...shit? There could be a 'vk' record or just anything at all.

Any idea what you do with those? I mean,...there must be reason behind this?

Reply

ERZ October 23, 2015 at 1:37 PM

as you have seen, an hbin is just a container. The order means nothing for any of the records. you have to look a the relative offset and then use that to jump to and from things. records start at offset 0x20, but not always nk records. =) it could be a vk, a data cell, a list, etc.

Reply

Enter your comment...

Comment as: ggyy (Google)

Sign out

Publish Preview

Notify me