PRTK / DNA

Dustin Hurlbut

For more information contact: info@syntricate.com

Three New Rule Options Available:

There are three new options that can be used to construct custom rules. The three are shown below in Figure 1. They also appear in the default rules available to be selected when creating an attack profile.

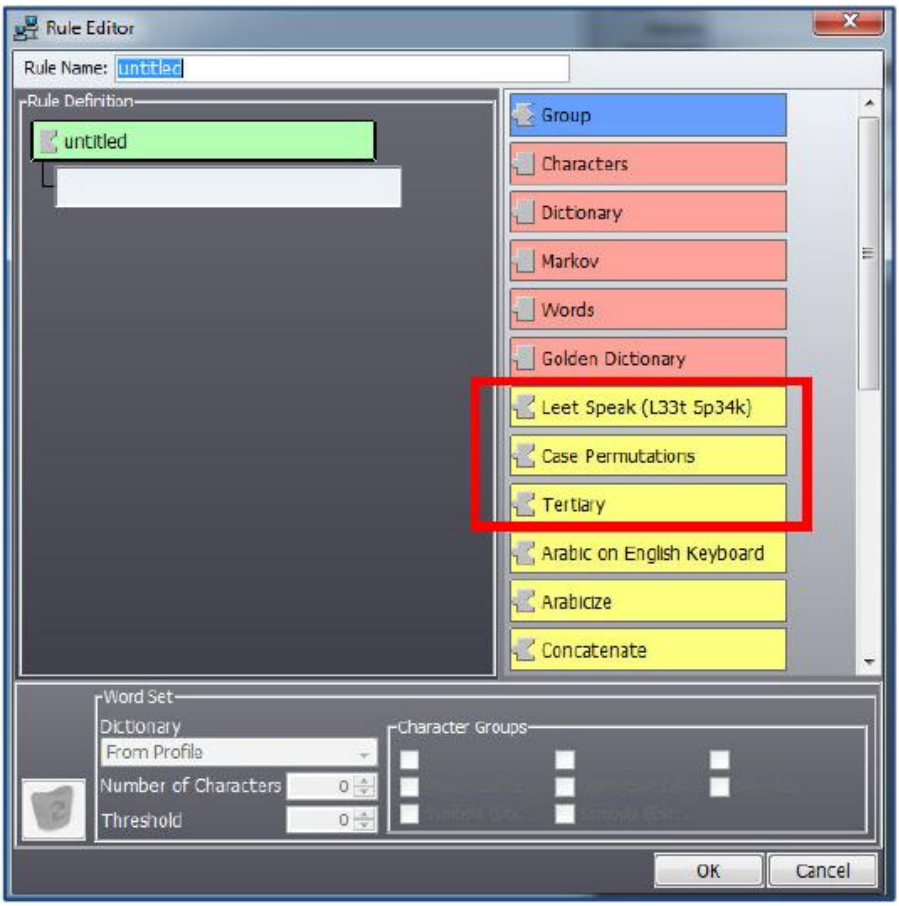

Figure 1 - New Custom Rule Selections

SYNTRICATE

PRTK / DNA

Dustin Hurlbut

For more information contact: info@syntricate.com

Leet Speak (L33t 5p34k)

Leetspeak, according to Wikipedia, is derived from the English word "elite" user. It is the substitution of ASCII characters for certain English alphabet characters, such as: password = p@ssword or = p@$$w0rd. We have commonly referred to this as character substitution in class.

Our previous rule for dealing with character substitution was BAS-2-20. This rule is still available to us and was listed as a Dictionary Primary Character Replacement Search. It would take each letter in a word and check for substitutions. However, it could only detect one substitution. For example; it could find "p@ssword" instead of password, but it could not find p@$$word as the "$" characters were a second substitution. This was obviously limiting our ability to crack this type of password.

The new rule for this is:

- ADV-1-29 - Multiple character replacements from selected dictionaries (Leet)

It will search for multiple substitutions, so it will find p@ssword, p@$$word, or p@$$w0rd for the dictionary word "password". This rule could get quite large so it should be used with careful consideration of the dictionaries it is associated with. A listing of the character substitutions made in AccessData products is in Appendix 1 at the end of this paper.

Case Permutations

Case Permutations is a method of changing the case of potential characters. It will go through a dictionary word and try all the possible permutations of upper and lower case. Theoretically, it will take the word "duck" and make the following permutations:

- duck
- Duck
- DUck
- DUCk
- DUCK
- dUCK

K

- K
- duCK
- ducK
- dUcK
- DuCk
- DucK
- dUCk

The new rule for case toggling is:

- ADV-1-28 - Toggles all upper/lower case using selected dictionaries

ADV-1-28 will get all the above permutations of "duck".

Tertiary

Tertiary refers to the third type of dictionary attack we now have at our disposal. Primary is the standard dictionary test that takes each word and runs four tests on it:

- duck
- DUCK
- Duck
- dUCK

Secondary makes two tests; all lower case and all upper case:

- duck
- DUCK

Tertiary makes two tests as well, but they are different than secondary tests; namely they toggle the case set of each word. If you have an As-Is Dictionary with the word "dUcK", the tertiary test will toggle that word to "DuCk". It also changes the word to title case and the rest lowercase. So, our word "dUcK" will also be changed to "Duck".

The new rule for tertiary is:

- ADV-1-30 - Toggles both letter cases and the title cases from selected dictionaries

Other Rules

There are two more new rules:

- BAS-3-08 - Lowercases words from selected dictionaries
- BAS-3-09 - Lowercases words from selected dictionaries followed by a '1'

They are both obvious in what they do by their name, namely BAS-3-08 sets all words in the dictionary to lowercase and BAS-3-09 does the same operation but sets a "1" behind each of the words.

Other Changes

As-Is Rule - We no longer have to create a custom As-Is dictionary rule to retain the case of word sets.

- BAS-3-10 - Uses entries "AS-IS" from selected dictionaries

Selecting BAS-3-10 will run every word in the dictionary only in the case they are set at. So a dictionary with the word "DucK" will only run that word as "DucK". In previous versions, a user had to make a custom rule to run an As-Is dictionary. This works nicely with the strings.exe command for retaining case in an index.

PRTK / DNA

Dustin Hurlbut

For more information contact: info@syntricate.com

Ability to Move Dictionary Order in Profiles - We can now set the order desired on processing dictionaries in the profile. Figure 2 and 3 show the new Profile's box. Note, there is a tab in the Dictionaries pane called Order.

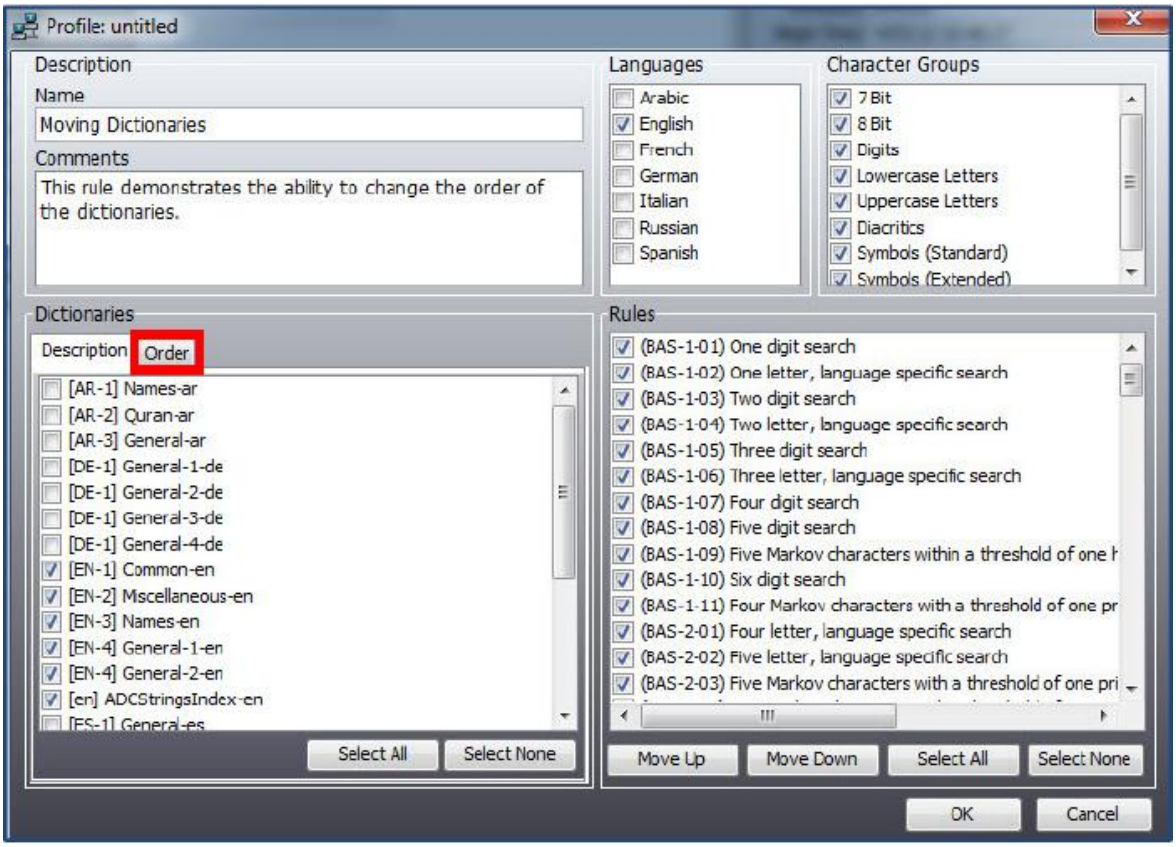

Figure 2 - Dictionaries Pane Changes

SYNTRICATE

In Figure 3, I have selected the Order tab and moved the dictionaries around. This will help ordering dictionaries in class to get at the ones you want to target first.

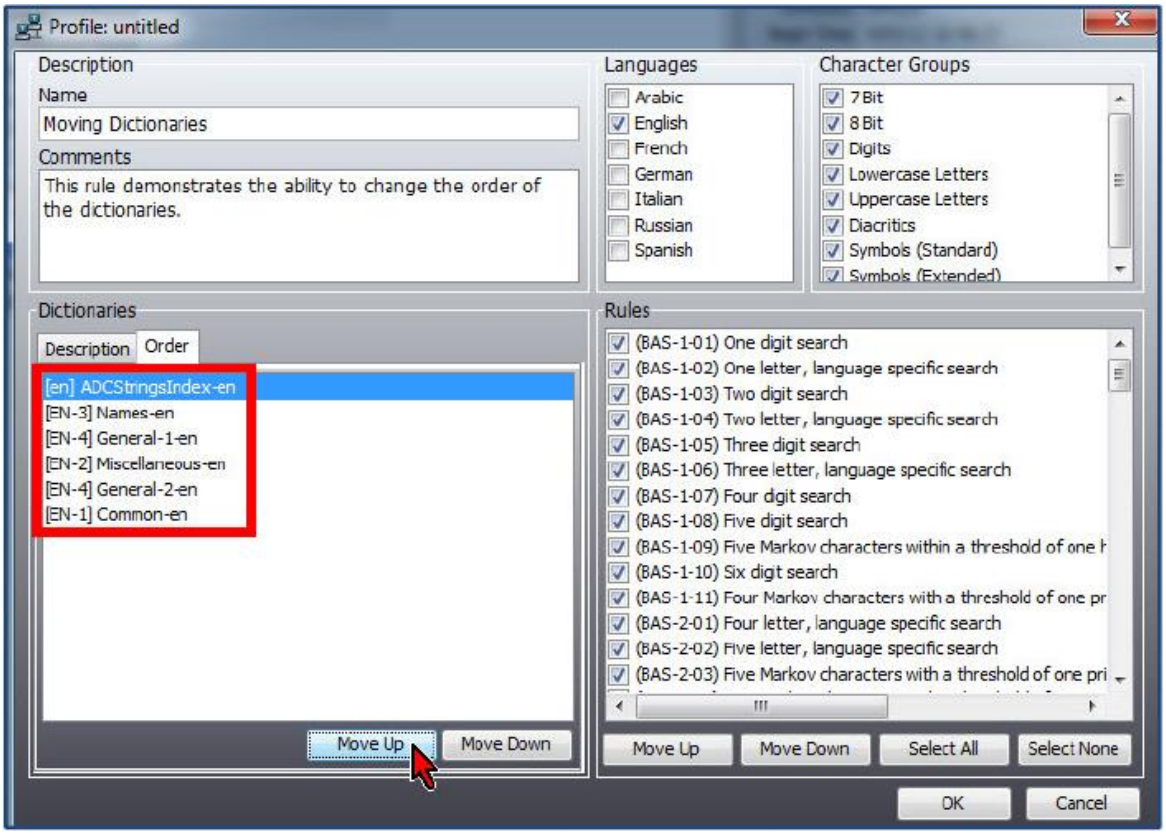

Figure 3 - Moving the Dictionary Order

Dictionary Selector - There is also an added feature to select dictionaries. If you have an index from a large image, it will make selecting multiple dictionaries much easier than having to select each one individually. You can now hold down the Shift + Alt key and click on one of the dictionaries. This will select all of the same named dictionaries along with their sequential numbers.

Dustin Hurlbut

For more information contact: info@syntricate.com

Worker Processing - There was a major change in how the workers are utilizing the resources of the CPU in the system. PRTK / DNA are now using the actual physical cores rather than the displayed cores. For example, in Figure 4, my system appears as though it has eight processors.

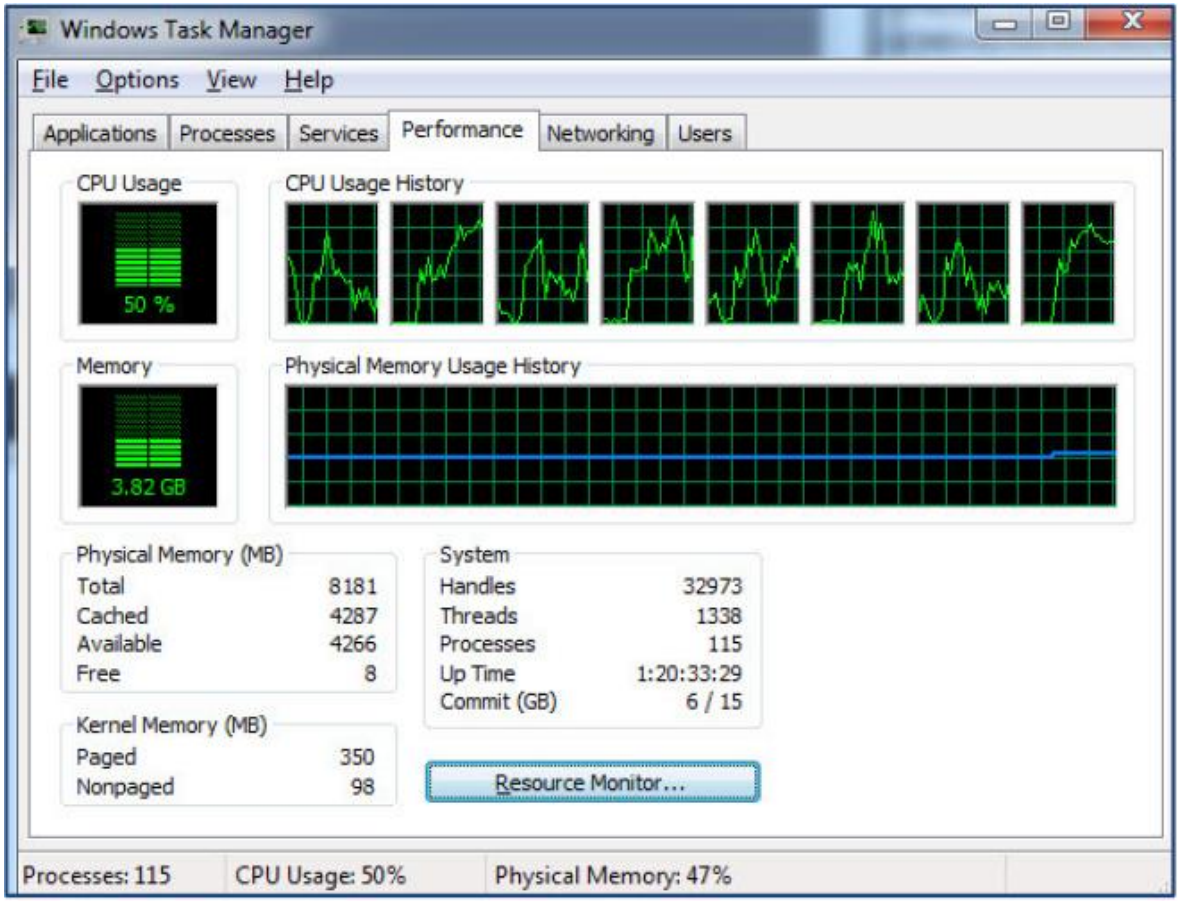

Figure 4 - Windows Task Manager Displaying Processing Job in DNA

This is a quad core I7 Intel processor. According to Harry Mahoney at AccessData, the Microsoft operating system displays the Hyper-Threading as another core. So in Figure 4, you see displayed eight cores rather than the actual four physical cores that exist in the system. When processing a job, it will appear that the system is running about half speed when we are actually running at 100% of the resources of the four physical cores.

The release notes indicate the processing has been improved, and it appears it has. In class, I usually see about 800,000 to 900,000 password tries per second with this machine on a Word 97/2000 Office document. In Figure 5, you can see that my performance is running at about one million, one hundred thousand with an Office 2000 document.

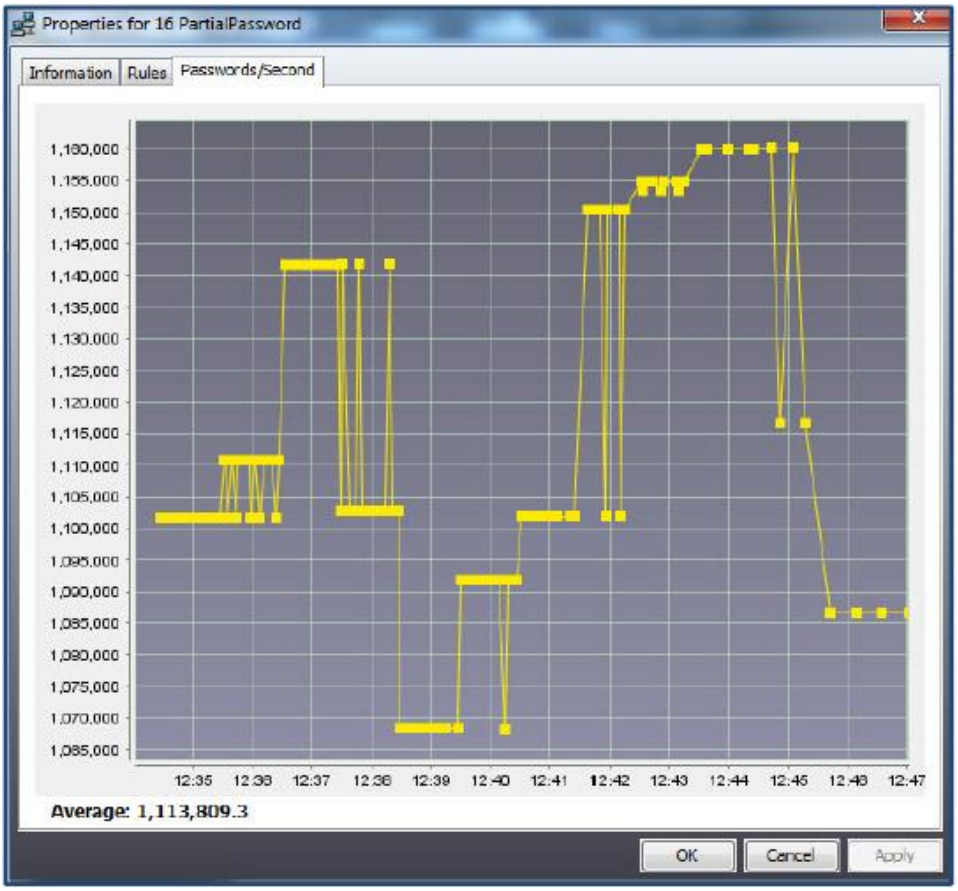

Figure 5 - Passwords Per Second in DNA - Microsoft Office 2000 Document

According to Harry, we assign a worker per physical core rather than one worker per core as done before. By allocating workers to "fake" Hyper-Threading cores as we did in the past, we could overload the CPU which resulted in slower processing and the potential to lock up the application. This actually improves performance and we are still able to work with the Hyper-Threading. We're just not pretending they are a separate core.

Dictionary Generator Default Settings - This doesn't appear to be documented, but they have changed the default settings of the Dictionary Generator. Figure 6 shows the new settings when you select the More Settings button. In the past, the Symbols checkbox was not selected by default. It is now selected by default.

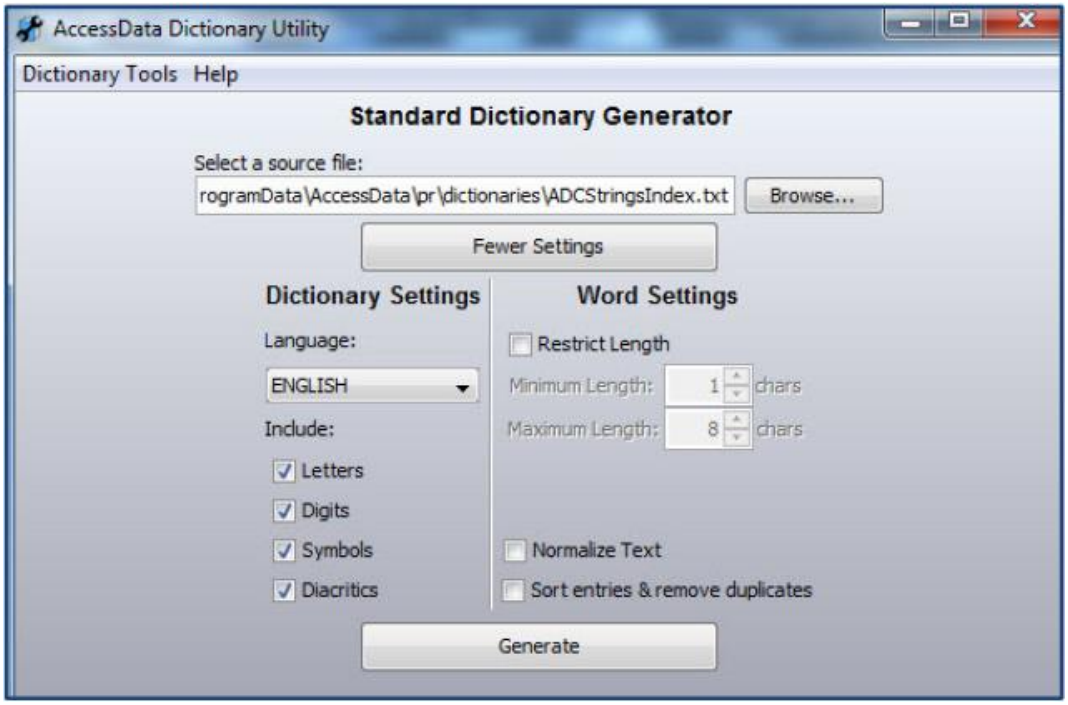

Figure 6 - Dictionary Generator Default Settings

Also note that the Normalize Text checkbox is not selected. In previous versions, this was selected by default.

Passphrase Generator - This was designed to generate passphrases from a body of text, like a Word document. In the past it ignored punctuation, because the Symbols checkbox was deselected by default. This version incorporates punctuation into the generated passphrase dictionary when the Symbols checkbox is selected. You'll have to be careful to turn this off before generating a passphrase dictionary from a document, if that document has punctuation in it.

PRTK / DNA

Dustin Hurlbut

For more information contact: info@syntricate.com

APPENDIX 1

-Leetspeak Substitution
-Leet does not mimic BAS-2-20

A: 4 @
B: 6 13
C: < ([
D: ?
E: 3 &
F: } ph
G: 6 9
H: #
I: ! | :
J:
K: |<
L: |
M:
N:
O: 0 ()
P: ? 9
Q: 9
R: 2
S: 5 $ z
T: 7 +
U:
V:
W:
X: % *
Y:
Z: 2

SYNTRICATE

Every possible substitution is made, so leet levels can be really huge.

For example, "dog" will become: (not necessarily in this order)
dog
?og
d0g
?0g
d()g
?()g
do9
?o9
d09
?09
d()9
?()9

Group Offsets

Group offsets are located at:

SAM\SAM\Domains\Builtin\Aliases\00000###

Example:

Subkey Name: 00000220

Note that the number in this example converts to 544. The numbers are in hex format to identify the particular group.

Offsets	Description	Comment
16–27	Pointer to group name	12-byte dataset
28–39	Pointer to group description	12-byte dataset
35–47	Pointer to group members	12-byte dataset

Divide the 12-byte dataset into three sets of four bytes each. The first four bytes is the pointer to the beginning of the designated data plus 204 bytes. The middle four bytes defines the size of the data. The last four bytes are not used. For example, the dataset 0x980000001c00000000000000, is divided up to:

- 0x98000000 = Pointer to the group name (0xbc = 152 + 52 = 204 as beginning offset)
- 0x1c000000 = Size of the data (28 bytes)
- 0x00000000 = Not used

User Assist Offsets

Offsets	Description	Comment
0-3	Session Number	
4-7	Use Count	Begins at 5 so first use will show a 6
8-15	Last launched date and time	64-bit Windows Date/Time Stamp