Accessing ACPI SLIC From UEFI Shell

Finnbarr P. Murphy
(fpm@fpmurphy.com)

Those who follow my work in the UEFI Shell space are aware that I usually develop a number of new, and hopefully useful, UEFI shell utilities each year. This year, I plan to write three or four new utilities and enhance a number of existing utilities. This is the first of these new utilities.

In this post, I describe the *ShowSLIC* utility. It is the first of my new utilities and came about from license and booting issues caused by a disk failure on a friend's laptop that was running Windows 7.

ShowSLIC is designed to enable you to retrieve *SLIC* (System License Internal Certificate) information from a UEFI-based Microsoft Windows PC or laptop. Such information is accessible (exposed) via the ACPI (Advanced Configuration and Power Interface) SLIP table.

Some history first. Licensing was easy to bypass (defeat) in Windows XP and earlier versions of Microsoft Windows. Windows Vista was the first version of Microsoft Windows whose royalty OEM licensing mechanism was harder to defeat. Basically, the BIOS was modified by an OEM to contain specific licensing information. This information was exposed to the operating system via the ACPI SLIC table.

When Windows Vista was installed by a royalty OEM, it would be given certificate files with appropriate OEM information and public keys. The certificates themselves were signed by a Microsoft private key to resist tampering. The Royalty OEM process was triggered by use of a generic OEM serial number (product key), common to all computers sold by a particular OEM. When the product key was entered, the system firmware was checked for a SLIC table, and the contents compared to any and all OEM certificates that had been loaded by Windows. If a SLIC table was present, and matched a valid OEM certificate, Windows Vista was offline activated.

The problem with this activation mechanism was that a royalty OEM generally had one key for each major version of Windows Vista (Home, Professional, etc), and you could change versions of Windows Vista by simply using a different OEM key.

Windows 7 OEM activation was exactly the same as for Windows Vista, just the certificate version was increased from SLIC 2.0 to SLIC 2.1. Windows 8 and Windows 8.1 could also use SLIC activation but the version of SLIC was rev'ed to SLIC 3.0.

Windows 8 was also the first version of Microsoft Windows to use a new royalty OEM licensing mechanism based on an ACPI table called *MSDM*. This table contains a hardware hash that matches the machine it is installed on, along with a full OEM Windows key, which is specific to the machine it is installed on. Activation via a generic OEM product key is no longer used, instead each OEM computer requires a unique activation product key. By the way, my *ShowMSDM* utility can be used to retrieve full MSDM information from the UEFI shell.

SLIC should not be confused with *SLP* (System Locked Pre-installation), which was a entirely different mechanism used by the major computer manufacturers to pre-activate Microsoft Windows computers before distribution. See this Wikipedia article for more information.

Here is the source code for the *ShowSLIC* utility:


```
//
//  Copyright (c) 2018  Finnbarr P. Murphy.   All rights reserved.
//
//  Display SLIC License Key
//
//  License: BSD License
//
#include <Uefi.h>
#include <Library/UefiLib.h>
#include <Library/ShellCEntryLib.h>
#include <Library/ShellLib.h>
#include <Library/BaseLib.h>
#include <Library/BaseMemoryLib.h>
#include <Library/UefiBootServicesTableLib.h>
#include <Library/PrintLib.h>
#include <Protocol/EfiShell.h>
#include <Protocol/LoadedImage.h>
#include <Protocol/AcpiSystemDescriptionTable.h>
#include <Guid/Acpi.h>
#define UTILITY_VERSION L"20180227"
#undef DEBUG
#pragma pack(1)
// OEM Public Key
typedef struct {
    UINT32    Type;
    UINT32    Length;
    UINT8     KeyType;
    UINT8     Version;
    UINT16    Reserved;
    UINT32    Algorithm;
    CHAR8     Magic[4];
    UINT32    BitLength;
    UINT32    Exponent;
    UINT8     Modulus[128];
} OEM_PUBLIC_KEY;
// Windows Marker
typedef struct {
    UINT32    Type;
    UINT32    Length;
    UINT32    Version;
    CHAR8     OemId[6];
    CHAR8     OemTableId[8];
    CHAR8     Product[8];
    UINT16    MinorVersion;
    UINT16    MajorVersion;
    UINT8     Signature[144];
} WINDOWS_MARKER;
// Software Licensing
typedef struct {
    EFI_ACPI_SDT_HEADER Header;
    OEM_PUBLIC_KEY      PubKey;
    WINDOWS_MARKER      Marker;
} EFI_ACPI_SLIC;
#pragma pack()
static VOID AsciiToUnicodeSize(CHAR8 *, UINT8, CHAR16 *, BOOLEAN);
static VOID
AsciiToUnicodeSize( CHAR8 *String,
                    UINT8 length,
                    CHAR16 *UniString,
                    BOOLEAN Quote )
{
    int len = length;
    if (Quote)
        *(UniString++) = L'"';
    while (*String != '&#92;&#48;' &amp;&amp; len > 0) {
        *(UniString++) = (CHAR16) *(String++);
```

```
        len--;
    }
    if (Quote)
        *(UniString++) = L'"';
    *UniString = '&amp;#92;&amp;#48;';
}
static VOID
PrintHexTable( UINT8 *ptr,
               int count )
{
    int i = 0;
    Print(L"  ");
    for (i = 0; i < count; i++ ) {
        if ( i > 0 &amp;amp;&amp;amp; i%16 == 0)
            Print(L"\n  ");
        Print(L"0x%02x ", 0xff &amp;amp; *ptr++);
    }
    Print(L"\n");
}
static VOID
PrintAcpiHeader( EFI_ACPI_SDT_HEADER *Ptr )
{
    CHAR16 Buffer[50];
    Print(L"ACPI Standard Header\n");
    AsciiToUnicodeSize((CHAR8 *)&amp;amp;(Ptr->Signature), 4, Buffer, TRUE);
    Print(L"  Signature        : %s\n", Buffer);
    Print(L"  Length           : 0x%08x (%d)\n", Ptr->Length, Ptr->Length);
    Print(L"  Revision         : 0x%02x (%d)\n", Ptr->Revision, Ptr->Revision);
    Print(L"  Checksum         : 0x%02x (%d)\n", Ptr->Checksum, Ptr->Checksum);
    AsciiToUnicodeSize((CHAR8 *)&amp;amp;(Ptr->OemId), 6, Buffer, TRUE);
    Print(L"  OEM ID           : %s\n", Buffer);
    AsciiToUnicodeSize((CHAR8 *)&amp;amp;(Ptr->OemTableId), 8, Buffer, TRUE);
    Print(L"  OEM Table ID     : %s\n", Buffer);
    Print(L"  OEM Revision     : 0x%08x (%d)\n", Ptr->OemRevision, Ptr->OemRevision);
    AsciiToUnicodeSize((CHAR8 *)&amp;amp;(Ptr->CreatorId), 4, Buffer, TRUE);
    Print(L"  Creator ID       : %s\n", Buffer);
    Print(L"  Creator Revision  : 0x%08x (%d)\n", Ptr->CreatorRevision, Ptr->CreatorRevisi
on);
    Print(L"\n");
}
static VOID
PrintOemPublicKey( OEM_PUBLIC_KEY *Ptr,
                   int Verbose )
{
    CHAR16 Buffer[50];
    Print(L"OEM Public Key\n");
    Print(L"  Type             : 0x%08x (%d)\n", Ptr->Type, Ptr->Type);
    Print(L"  Length           : 0x%08x (%d)\n", Ptr->Length, Ptr->Length);
    Print(L"  KeyType          : 0x%02x (%d)\n", Ptr->KeyType, Ptr->KeyType);
    Print(L"  Version          : 0x%02x (%d)\n", Ptr->Version, Ptr->Version);
    Print(L"  Reserved         : 0x%04x (%d)\n", Ptr->Reserved, Ptr->Reserved);
    Print(L"  Algorithm        : 0x%08x (%d)\n", Ptr->Algorithm, Ptr->Algorithm);
    AsciiToUnicodeSize((CHAR8 *)&amp;amp;(Ptr->Magic), 4, Buffer, TRUE);
    Print(L"  Magic            : %s\n", Buffer);
    Print(L"  Bit Length       : 0x%08x (%d)\n", Ptr->BitLength, Ptr->BitLength);
    Print(L"  Exponent         : 0x%08x (%d)\n", Ptr->Exponent, Ptr->Exponent);
    if (Verbose) {
        Print(L"  Modulus:\n");
        PrintHexTable( (UINT8 *)&amp;amp;(Ptr->Modulus), (int)(Ptr->BitLength)/8 );
    }
    Print(L"\n");
}
static VOID
PrintWindowsMarker( WINDOWS_MARKER *Ptr,
                    int Verbose )
{
    CHAR16 Buffer[50];
    Print(L"Windows Marker\n");
```



```
    Print(L"  Type              : 0x%08x (%d)\n", Ptr->Type, Ptr->Type);
    Print(L"  Length            : 0x%08x (%d)\n", Ptr->Length, Ptr->Length);
    Print(L"  Version           : 0x%02x (%d)\n", Ptr->Version, Ptr->Version);
    AsciiToUnicodeSize((CHAR8 *)(Ptr->OemId), 6, Buffer, TRUE);
    Print(L"  OEM ID            : %s\n", Buffer);
    AsciiToUnicodeSize((CHAR8 *)(Ptr->OemTableId), 8, Buffer, TRUE);
    Print(L"  OEM Table ID      : %s\n", Buffer);
    AsciiToUnicodeSize((CHAR8 *)(Ptr->Product), 8, Buffer, TRUE);
    Print(L"  Windows Flag      : %s\n", Buffer);
    Print(L"  SLIC Version      : 0x%04x%04x (%d.%d)\n", Ptr->MajorVersion, Ptr->MinorVers
ion,
                                                         Ptr->MajorVersion, Ptr->MinorVers
ion);
    if (Verbose) {
        Print(L"  Signature:\n");
        PrintHexTable( (UINT8 *)&amp;(Ptr->Signature), 144 );   // 144 - Fix up in next ve
rsion
    }
    Print(L"\n");
}
static VOID
PrintSLIC( EFI_ACPI_SLIC *Slic,
           int Verbose )
{
    Print(L"\n");
    PrintAcpiHeader( (EFI_ACPI_SDT_HEADER *)&amp;(Slic->Header) );
    PrintOemPublicKey( (OEM_PUBLIC_KEY *)&amp;(Slic->PubKey), Verbose );
    PrintWindowsMarker( (WINDOWS_MARKER *)&amp;(Slic->Marker), Verbose );
}
static int
ParseRSDP( EFI_ACPI_2_0_ROOT_SYSTEM_DESCRIPTION_POINTER *Rsdp,
           CHAR16* GuidStr,
           int Verbose )
{
    EFI_ACPI_SDT_HEADER *Xsdt, *Entry;
#ifdef DEBUG
    CHAR16 OemStr[20];
#endif
    UINT32 EntryCount;
    UINT64 *EntryPtr;
#ifdef DEBUG
    Print(L"\n\nACPI GUID: %s\n", GuidStr);
    AsciiToUnicodeSize((CHAR8 *)(Rsdp->OemId), 6, OemStr, FALSE);
    Print(L"\nFound RSDP. Version: %d  OEM ID: %s\n", (int)(Rsdp->Revision), OemStr);
#endif
    if (Rsdp->Revision >= EFI_ACPI_2_0_ROOT_SYSTEM_DESCRIPTION_POINTER_REVISION) {
        Xsdt = (EFI_ACPI_SDT_HEADER *)(Rsdp->XsdtAddress);
    } else {
#ifdef DEBUG
        Print(L"ERROR: No ACPI XSDT table found.\n");
#endif
        return 1;
    }
    if (Xsdt->Signature != SIGNATURE_32 ('X', 'S', 'D', 'T')) {
#ifdef DEBUG
        Print(L"ERROR: Invalid ACPI XSDT table found.\n");
#endif
        return 1;
    }
    EntryCount = (Xsdt->Length - sizeof (EFI_ACPI_SDT_HEADER)) / sizeof(UINT64);
#ifdef DEBUG
    AsciiToUnicodeSize((CHAR8 *)(Xsdt->OemId), 6, OemStr, FALSE);
    Print(L"Found XSDT. OEM ID: %s  Entry Count: %d\n\n", OemStr, EntryCount);
#endif
    EntryPtr = (UINT64 *)(Xsdt + 1);
    for (int Index = 0; Index < EntryCount; Index++, EntryPtr++) {
        Entry = (EFI_ACPI_SDT_HEADER *)((UINTN)(*EntryPtr));
        if (Entry->Signature == SIGNATURE_32 ('S', 'L', 'I', 'C')) {
```



```
            PrintSLIC((EFI_ACPI_SLIC *)((UINTN)(*EntryPtr)), Verbose);
        }
    }
    return 0;
}
static void
Usage( void )
{
    Print(L"Usage: ShowSLIC [-v | --verbose]\n");
    Print(L"       ShowSLIC [-V | --version]\n");
}
INTN
EFIAPI
ShellAppMain( UINTN Argc,
              CHAR16 **Argv )
{
    EFI_CONFIGURATION_TABLE *ect = gST->ConfigurationTable;
    EFI_ACPI_2_0_ROOT_SYSTEM_DESCRIPTION_POINTER *Rsdp = NULL;
    EFI_GUID gAcpi20TableGuid = EFI_ACPI_20_TABLE_GUID;
    EFI_GUID gAcpi10TableGuid = ACPI_10_TABLE_GUID;
    EFI_STATUS Status = EFI_SUCCESS;
    CHAR16 GuidStr[100];
    int Verbose = 0;
    if (Argc == 2) {
        if (!StrCmp(Argv[1], L"--verbose") ||
            !StrCmp(Argv[1], L"-v")) {
            Verbose = 1;
        } else if (!StrCmp(Argv[1], L"--version") ||
            !StrCmp(Argv[1], L"-V")) {
            Print(L"Version: %s\n", UTILITY_VERSION);
            return Status;
        } else if (!StrCmp(Argv[1], L"--help") ||
            !StrCmp(Argv[1], L"-h")) {
            Usage();
            return Status;
        } else {
            Usage();
            return Status;
        }
    }
    if (Argc > 2) {
        Usage();
        return Status;
    }
    // locate RSDP (Root System Description Pointer)
    for (int i = 0; i < gST->NumberOfTableEntries; i++) {
        if ((CompareGuid (&amp;(gST->ConfigurationTable[i].VendorGuid), &amp;gAcpi20TableG
uid)) ||
            (CompareGuid (&amp;(gST->ConfigurationTable[i].VendorGuid), &amp;gAcpi10TableG
uid))) {
            if (!AsciiStrnCmp("RSD PTR ", (CHAR8 *)(ect->VendorTable), 8)) {
                UnicodeSPrint(GuidStr, sizeof(GuidStr), L"%g", &amp;(gST->ConfigurationTab
le[i].VendorGuid));
                Rsdp = (EFI_ACPI_2_0_ROOT_SYSTEM_DESCRIPTION_POINTER *)ect->VendorTable;
                ParseRSDP(Rsdp, GuidStr, Verbose);
            }
        }
        ect++;
    }
    if (Rsdp == NULL) {
        if (Verbose) {
            Print(L"ERROR: Could not find an ACPI RSDP table.\n");
        }
        Status = EFI_NOT_FOUND;
    }
    return Status;
}
```


There is nothing unusual in the above code so I will not go into details about each function. It was designed to be built using the *UDK2017* snapshot of TianoCore EDK2.

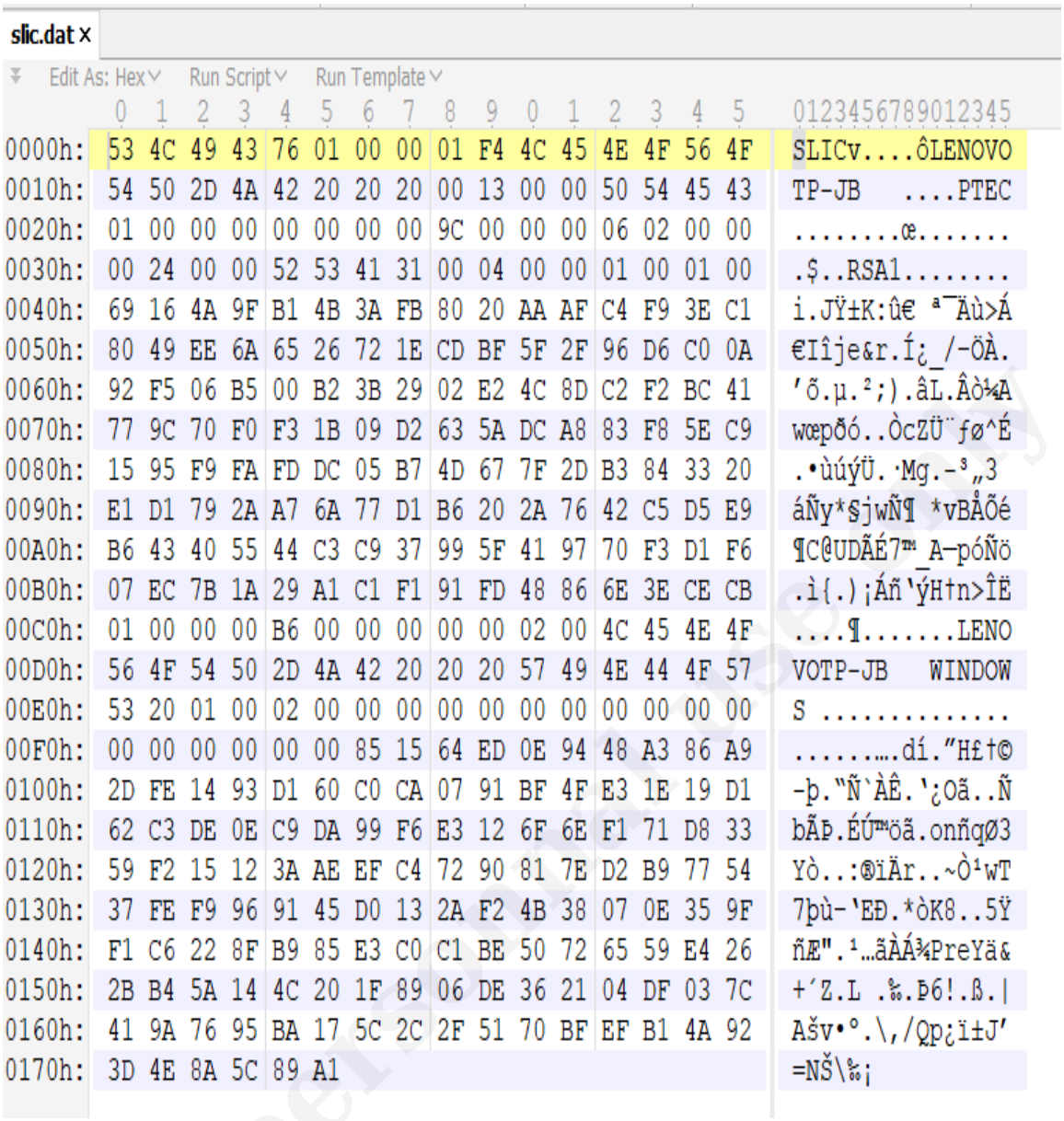
Note that I have switched all UEFI utilities that I maintain from *GNU-EFI or UDK2015* and GCC to *UDK2017* and LLVM Clang. Note also that I no longer use utility version numbers such as "1.0" or "1.1". Now I use a date string of the form "YYYYMMDD".

Here is the *INF* build configuration file used to build this utility:

```
[Defines]
  INF_VERSION                    = 0x00010006
  BASE_NAME                      = ShowSLIC
  FILE_GUID                      = 4ea87c51-7395-4dcd-0055-747010f3ce51
  MODULE_TYPE                    = UEFI_APPLICATION
  VERSION_STRING                 = 0.1
  ENTRY_POINT                    = ShellCEntryLib
  VALID_ARCHITECTURES            = X64

[Sources]
  ShowSLIC.c

[Packages]
  MdePkg/MdePkg.dec
  ShellPkg/ShellPkg.dec

[LibraryClasses]
  ShellCEntryLib
  ShellLib
  BaseLib
  BaseMemoryLib
  UefiLib

[Protocols]

[BuildOptions]

[Pcd]
```


AS usual, I expect that you to know what an *.INF* is if you are reading this post so no further build instructions are provided.

Here is a binary dump of the SLIC table from a Lenovo T450 laptop which originally came with Windows 7, was upgraded to Windows 8, followed by Windows 8.1 and now runs Windows 10:



slic.dat x

Edit As: Hex Run Script Run Template

	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	0123456789012345
0000h:	53	4C	49	43	76	01	00	00	01	F4	4C	45	4E	4F	56	4F	SLICv....ôLENOVO
0010h:	54	50	2D	4A	42	20	20	20	00	13	00	00	50	54	45	43	TP-JB ....PTEC
0020h:	01	00	00	00	00	00	00	00	9C	00	00	00	06	02	00	00	........œ.....
0030h:	00	24	00	00	52	53	41	31	00	04	00	00	01	00	01	00	.$..RSA1........
0040h:	69	16	4A	9F	B1	4B	3A	FB	80	20	AA	AF	C4	F9	3E	C1	i.JŸ±K:û€ ª¯Äù>Á
0050h:	80	49	EE	6A	65	26	72	1E	CD	BF	5F	2F	96	D6	C0	0A	€Iîje&r.Í¿_/-ÖÀ.
0060h:	92	F5	06	B5	00	B2	3B	29	02	E2	4C	8D	C2	F2	BC	41	'õ.µ.²;).âL.Âò¼A
0070h:	77	9C	70	F0	F3	1B	09	D2	63	5A	DC	A8	83	F8	5E	C9	wœpðó..ÒcZÜ¨ƒø^É
0080h:	15	95	F9	FA	FD	DC	05	B7	4D	67	7F	2D	B3	84	33	20	.•ùúýÜ.·Mg.-³„3
0090h:	E1	D1	79	2A	A7	6A	77	D1	B6	20	2A	76	42	C5	D5	E9	áÑy*§jwÑ¶ *vBÅÕé
00A0h:	B6	43	40	55	44	C3	C9	37	99	5F	41	97	70	F3	D1	F6	¶C@UDÃÉ7™_A—póÑö
00B0h:	07	EC	7B	1A	29	A1	C1	F1	91	FD	48	86	6E	3E	CE	CB	.ì{.)¡Áñ'ýH†n>ÎË
00C0h:	01	00	00	00	B6	00	00	00	00	00	02	00	4C	45	4E	4F	....¶.......LENO
00D0h:	56	4F	54	50	2D	4A	42	20	20	20	57	49	4E	44	4F	57	VOTP-JB WINDOW
00E0h:	53	20	01	00	02	00	00	00	00	00	00	00	00	00	00	00	S .....
00F0h:	00	00	00	00	00	00	85	15	64	ED	0E	94	48	A3	86	A9	.....….dí.”H£†©
0100h:	2D	FE	14	93	D1	60	C0	CA	07	91	BF	4F	E3	1E	19	D1	-þ.“Ñ`ÀÊ.'¿Oã..Ñ
0110h:	62	C3	DE	0E	C9	DA	99	F6	E3	12	6F	6E	F1	71	D8	33	bÃÞ.ÉÚ™öã.onñqØ3
0120h:	59	F2	15	12	3A	AE	EF	C4	72	90	81	7E	D2	B9	77	54	Yò..:®ïÄr..~Ò¹wT
0130h:	37	FE	F9	96	91	45	D0	13	2A	F2	4B	38	07	0E	35	9F	7þù-'EÐ.*òK8..5Ÿ
0140h:	F1	C6	22	8F	B9	85	E3	C0	C1	BE	50	72	65	59	E4	26	ñÆ".¹…ãÀÁ¾PreYä&
0150h:	2B	B4	5A	14	4C	20	1F	89	06	DE	36	21	04	DF	03	7C	+´Z.L .‰.Þ6!.ß.|
0160h:	41	9A	76	95	BA	17	5C	2C	2F	51	70	BF	EF	B1	4A	92	Ašv•°.\,/Qp¿ï±J'
0170h:	3D	4E	8A	5C	89	A1											=NŠ\‰¡

Here is the output from this utility when it was run on this laptop:

```
fs1> ShowSLIC.efi -v

ACPI Standard Header
  Signature         : "SLIC"
  Length            : 0x00000176 (374)
  Revision          : 0x01 (1)
  Checksum          : 0xF4 (244)
  OEM ID            : "LENOVO"
  OEM Table ID      : "TP-JB   "
  OEM Revision      : 0x00001300 (4864)
  Creator ID        : "PTEC"
  Creator Revision  : 0x00000001 (1)

OEM Public Key
  Type              : 0x00000000 (0)
  Length            : 0x0000009C (156)
  KeyType           : 0x06 (6)
  Version           : 0x02 (2)
  Reserved          : 0x0000 (0)
  Algorithm         : 0x00002400 (9216)
```



```
  Magic             : "RSA1"
  Bit Length        : 0x00000400 (1024)
  Exponent          : 0x00010001 (65537)
  Modulus:
  0x69 0x16 0x4A 0x9F 0xB1 0x4B 0x3A 0xFB 0x80 0x20 0xAA 0xAF 0xC4 0xF9 0x3E 0xC1
  0x80 0x49 0xEE 0x6A 0x65 0x26 0x72 0x1E 0xCD 0xBF 0x5F 0x2F 0x96 0xD6 0xC0 0x0A
  0x92 0xF5 0x06 0xB5 0x00 0xB2 0x3B 0x29 0x02 0xE2 0x4C 0x8D 0xC2 0xF2 0xBC 0x41
  0x77 0x9C 0x70 0xF0 0xF3 0x1B 0x09 0xD2 0x63 0x5A 0xDC 0xA8 0x83 0xF8 0x5E 0xC9
  0x15 0x95 0xF9 0xFA 0xFD 0xDC 0x05 0xB7 0x4D 0x67 0x7F 0x2D 0xB3 0x84 0x33 0x20
  0xE1 0xD1 0x79 0x2A 0xA7 0x6A 0x77 0xD1 0xB6 0x20 0x2A 0x76 0x42 0xC5 0xD5 0xE9
  0xB6 0x43 0x40 0x55 0x44 0xC3 0xC9 0x37 0x99 0x5F 0x41 0x97 0x70 0xF3 0xD1 0xF6
  0x07 0xEC 0x7B 0x1A 0x29 0xA1 0xC1 0xF1 0x91 0xFD 0x48 0x86 0x6E 0x3E 0xCE 0xCB

Windows Marker
  Type              : 0x00000001 (1)
  Length            : 0x000000B6 (182)
  Version           : 0x20000 (131072)
  OEM ID            : "LENOVO"
  OEM Table ID      : "TP-JB   "
  Windows Flag      : "WINDOWS "
  SLIC Version      : 0x00020001 (2.1)
  Signature:
  0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
  0x85 0x15 0x64 0xED 0x0E 0x94 0x48 0xA3 0x86 0xA9 0x2D 0xFE 0x14 0x93 0xD1 0x60
  0xC0 0xCA 0x07 0x91 0xBF 0x4F 0xE3 0x1E 0x19 0xD1 0x62 0xC3 0xDE 0x0E 0xC9 0xDA
  0x99 0xF6 0xE3 0x12 0x6F 0x6E 0xF1 0x71 0xD8 0x33 0x59 0xF2 0x15 0x12 0x3A 0xAE
  0xEF 0xC4 0x72 0x90 0x81 0x7E 0xD2 0xB9 0x77 0x54 0x37 0xFE 0xF9 0x96 0x91 0x45
  0xD0 0x13 0x2A 0xF2 0x4B 0x38 0x07 0x0E 0x35 0x9F 0xF1 0xC6 0x22 0x8F 0xB9 0x85
  0xE3 0xC0 0xC1 0xBE 0x50 0x72 0x65 0x59 0xE4 0x26 0x2B 0xB4 0x5A 0x14 0x4C 0x20
  0x1F 0x89 0x06 0xDE 0x36 0x21 0x04 0xDF 0x03 0x7C 0x41 0x9A 0x76 0x95 0xBA 0x17
  0x5C 0x2C 0x2F 0x51 0x70 0xBF 0xEF 0xB1 0x4A 0x92 0x3D 0x4E 0x8A 0x5C 0x89 0xA1

fs1>
```

Note that *OEMID* and *OEMTableID* are not just exposed in the SLIC table. In fact, they are exposed in all system ACPI table headers except FACS and are considered the identifier of a category of physical computers. However, they play an important role in SLIC-based license activation as they are part of the *Windows Marker* section of a SLIC table.

Finally, I do not know how to extract information from the *signature* field of the Windows Marker section of the SLIC table. I do know that it is some sort of digital certificate generated by the appropriate version (in this case 2.1) of the Microsoft *OATool* (OEM Activation Tool).

Enjoy!

