Accessing ACPI MSDM From UEFI Shell

Finnbarr P. Murphy

(fpm@fpmurphy.com)

This post is an update on my February 2015 post on how to retrieve your Microsoft Windows 8 or later product key from your firmware via the ACPI MSDM (Microsoft Data Management) table and the UEFI Shell. You should read that post if you are unfamiliar with the ACPI MSDM table. Of note is the fact that the ACPI specification is now managed by the UEFI Forum.

The original version of this utility was named *ListMSDM.efi* and used Nigel Croxon's excellent GNI-EFI library. This version is named *ShowMSDM.efi* and is based on the *UDK2017* snapshot of the TianoCore EDK2 (EFI Development Kit, Version 2.)

Here is the current source code for *ShowMSDM.efi*:

```
//
//  Copyright (c) 2014-2018  Finnbarr P. Murphy.  All rights reserved.
//
//  Display ACPI MSDM and/or Microsoft Windows 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"20180221"
#undef DEBUG
#pragma pack(1)
typedef struct {
    UINT32   Version;
    UINT32   Reserved;
    UINT32   DataType;
    UINT32   DataReserved;
    UINT32   DataLength;
    CHAR8    Data[30];
} SOFTWARE_LICENSING;
// Microsoft Data Management table structure
typedef struct {
    EFI_ACPI_SDT_HEADER Header;
    SOFTWARE_LICENSING  SoftLic;
} EFI_ACPI_MSDM;
#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 = '&#92;&#48;';
}
static VOID
PrintHexTable( UINT8 *ptr,
               int Count )
{
    int i = 0;
    Print(L"  ");
    for (i = 0; i < Count; i++ ) {
        if ( i > 0 &amp;&amp; i%16 == 0)
            Print(L"\n  ");
        Print(L"0x%02x ", 0xff &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;(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;(Ptr->OemId), 6, Buffer, TRUE);
    Print(L"  OEM ID            : %s\n", Buffer);
    AsciiToUnicodeSize((CHAR8 *)&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;(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
PrintSoftwareLicensing( SOFTWARE_LICENSING *Ptr,
                        BOOLEAN Verbose )
{
    CHAR16 Buffer[50];
    if (Verbose) {
        Print(L"Software Licensing\n");
        Print(L"  Version       : 0x%08x (%d)\n", Ptr->Version, Ptr->Version);
        Print(L"  Reserved      : 0x%08x (%d)\n", Ptr->Reserved, Ptr->Reserved);
        Print(L"  Data Type     : 0x%08x (%d)\n", Ptr->DataType, Ptr->DataType);
        Print(L"  Data Reserved : 0x%08x (%d)\n", Ptr->DataReserved, Ptr->DataReserved);
        Print(L"  Data Length   : 0x%08x (%d)\n", Ptr->DataLength, Ptr->DataLength);
        AsciiToUnicodeSize((CHAR8 *)(Ptr->Data), 30, Buffer, TRUE);
        Print(L"  Data          : %s\n", Buffer);
    } else {
        AsciiToUnicodeSize((CHAR8 *)(Ptr->Data), 30, Buffer, FALSE);
        Print(L"  %s\n", Buffer);
    }
}
static VOID
PrintMSDM( EFI_ACPI_MSDM *Msdm,
           BOOLEAN Verbose,
```



```
           BOOLEAN Hexdump )
{
    Print(L"\n");
    if (Hexdump) {
        PrintHexTable( (UINT8 *)Msdm, (int)(Msdm->Header.Length) );
    } else {
        if (Verbose) {
            PrintAcpiHeader( (EFI_ACPI_SDT_HEADER *)&amp;amp;(Msdm->Header) );
        }
        PrintSoftwareLicensing( (SOFTWARE_LICENSING *)&amp;amp;(Msdm->SoftLic), Verbose);
    }
    Print(L"\n");
}
static int
ParseRSDP( EFI_ACPI_2_0_ROOT_SYSTEM_DESCRIPTION_POINTER *Rsdp,
           CHAR16* GuidStr,
           BOOLEAN Verbose,
           BOOLEAN Hexdump )
{
    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 ('M', 'S', 'D', 'M')) {
            PrintMSDM((EFI_ACPI_MSDM *)((UINTN)(*EntryPtr)), Verbose, Hexdump);
        }
    }
    return 0;
}
static void
Usage( void )
{
    Print(L"Usage: ShowMSDM [-v | --verbose]\n");
    Print(L"       ShowMSDM [-V | --version]\n");
    Print(L"       ShowMSDM [-d | --dump]\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];
    BOOLEAN Verbose = FALSE;
    BOOLEAN Hexdump = FALSE;
    if (Argc == 2) {
        if (!StrCmp(Argv[1], L"--verbose") ||
            !StrCmp(Argv[1], L"-v")) {
            Verbose = TRUE;
        } else if (!StrCmp(Argv[1], L"--dump") ||
            !StrCmp(Argv[1], L"-d")) {
            Hexdump = TRUE;
        } 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, Hexdump);
            }
        }
        ect++;
    }
    if (Rsdp == NULL) {
        if (Verbose) {
            Print(L"ERROR: Could not find an ACPI RSDP table.\n");
        }
        Status = EFI_NOT_FOUND;
    }
    return Status;
}
```

I assume you are familiar with ACPI tables, EDK2 and UEFI APIs if you are reading this post and thus make no attempt to explain the relatively simple source code.

Here is a suitable *.INF* build file for this utility:

```
[Defines]
```



```
  INF_VERSION                    = 1.25
  BASE_NAME                      = ShowMSDM
  FILE_GUID                      = 4ea87c51-7395-4dcd-0055-747010f3ce51
  MODULE_TYPE                    = UEFI_APPLICATION
  VERSION_STRING                 = 1.0
  ENTRY_POINT                    = ShellCEntryLib
  VALID_ARCHITECTURES            = X64

[Sources]
  ShowMSDM.c

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

[LibraryClasses]
  ShellCEntryLib
  ShellLib
  BaseLib
  BaseMemoryLib
  UefiLib

[Protocols]

[BuildOptions]

[Pcd]
```

Here is sample output from this utility:

```
fs1:> ShowMSDM.efi -h
Usage: ShowMSDM [-v | --verbose]
       ShowMSDM [-V | --version]
       ShowMSDM [-d | --dump]

fs1:> ShowMSDM.efi

   NJCD6-T6TDQ-WRYF7-GY44B-VQVQ2

fs1:> ShowMSDM.efi -v

ACPI Standard Header
  Signature        : "MSDM"
  Length           : 0x00000055 (85)
  Revision         : 0x03 (3)
  Checksum         : 0xBF (191)
  OEM ID           : "LENOVO"
  OEM Table ID     : "TP-JB   "
  OEM Revision     : 0x00001300 (4864)
  Creator ID       : "PTEC"
  Creator Revision  : 0x00000002 (2)

Software Licensing
  Version       : 0x00000001 (1)
  Reserved      : 0x00000000 (0)
  Data Type     : 0x00000001 (1)
  Data Reserved : 0x00000000 (0)
  Data Length   : 0x0000001D (29)
  Data          : "NJCD6-T6TDQ-WRYF7-GY44B-VQVQ2 "

fs1:> ShowMSDM.efi -d

  0x4D 0x53 0x44 0x4D 0x55 0x00 0x00 0x00 0x03 0xBF 0x4C 0x45 0x4E 0x4F 0x56 0x4F
  0x54 0x50 0x2D 0x4A 0x42 0x20 0x20 0x20 0x00 0x13 0x00 0x00 0x50 0x54 0x45 0x43
  0x02 0x00 0x00 0x00 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x01 0x00 0x00 0x00
  0x00 0x00 0x00 0x00 0x1D 0x00 0x00 0x00 0x4E 0x4A 0x43 0x46 0x37 0x2D 0x54 0x36
```



```
  0x54 0x44 0x51 0x2D 0x57 0x52 0x59 0x46 0x37 0x2D 0x47 0x59 0x34 0x34 0x42 0x2D
  0x56 0x51 0x56 0x51 0x32

fs1:> ShowMSDM.efi -V
Version: 20180221

fs1:>
```

Note that the displayed product key is a made-up (fake) product key – so do not bother trying to use it.

Enjoy!

