So you are at your UEFI shell prompt and wish to display your UEFI firmware version and related information such as the vendor, revision and build date. How do you get this information? The `ver` command does not provide this information. You can parse the output of `smbiobview -t 0` and get the information.

Or you can simply compile and run the following UEFI utility:

```c
#include <efi.h>          // Install GNU_EFI package to get these headers
#define GUID EFI_GUID     // hack for SmBios.h
#include "SmBios.h"       // from EDK2. GNU_EFI libsmbios.h is defective
#include <efilib.h>
#define BUFSIZE 64
#define INVALID_HANDLE  (UINT16) (-1)
#define DMI_INVALID_HANDLE  0x83
#define DMI_SUCCESS 0x00
STATIC SMBIOS_TABLE_ENTRY_POINT *mSmbiosTable = NULL;
STATIC SMBIOS_STRUCTURE_POINTER m_SmbiosStruct;
STATIC SMBIOS_STRUCTURE_POINTER *mSmbiosStruct = &m_SmbiosStruct;
int Strlen(const char *str) {
    const char *s;
    for (s = str; *s; ++s);
    return (s - str);
}

CHAR8*
LibGetSmbiosString ( SMBIOS_STRUCTURE_POINTER *Smbios, UINT16 StringNumber) {
    UINT16 Index;
    CHAR8 *String;
    ASSERT (Smbios != NULL);
    String = (CHAR8 *) (Smbios->Raw + Smbios->Hdr->Length);
    for (Index = 1; Index <= StringNumber; Index++) {
        if (StringNumber == Index) {
            return String;
        }
        for (; *String != 0; String++) {
            String++;
        }
    }
    if (*String == 0) {
```
Display UEFI Firmware Version, Vendor, Revision and Build Date

EFI_STATUS
LibGetSmbiosStructure ( UINT16 *Handle,
    UINT8 **Buffer,
    UINT16 *Length)
{
    SMBIOS_STRUCTURE_POINTER Smbios;
    SMBIOS_STRUCTURE_POINTER SmbiosEnd;
    UINT8 *Raw;
    if (*Handle == INVALID_HANDLE) {
        *Handle = mSmbiosStruct->Hdr->Handle;
        return DMI_INVALID_HANDLE;
    }
    if ((Buffer == NULL) || (Length == NULL)) {
        Print(L"Invalid handle\n");
        return DMI_INVALID_HANDLE;
    }
    *Length = 0;
    Smbios.Hdr = mSmbiosStruct->Hdr;
    SmbiosEnd.Raw = Smbios.Raw + mSmbiosTable->TableLength;
    while (Smbios.Raw < SmbiosEnd.Raw) {
        if (Smbios.Hdr->Handle == *Handle) {
            Raw = Smbios.Raw;
            LibGetSmbiosString (&Smbios, (UINT16) (-1));
            *Length = (UINT16) (Smbios.Raw - Raw);
            *Buffer = Raw;
            if (Smbios.Raw < SmbiosEnd.Raw) {
                *Handle = Smbios.Hdr->Handle;
            } else {
                *Handle = INVALID_HANDLE;
            }
            return DMI_SUCCESS;
        }
        LibGetSmbiosString (&Smbios, (UINT16) (-1));
    }
    *Handle = INVALID_HANDLE;
    return DMI_INVALID_HANDLE;
}

CHAR16 *
ASCII_to_UCS2(const char *s, int len)
{
    CHAR16 *ret = NULL;
    int i;
    ret = AllocateZeroPool(len*2 + 2);
    if (!ret)
        return NULL;
    for (i = 0; i < len; i++)
        ret[i] = s[i];
    return ret;
}

EFI_STATUS
efi_main (EFI_HANDLE image, EFI_SYSTEM_TABLE *systab)
{
    EFI_STATUS status = EFI_SUCCESS;
    EFI_GUID Guid = EFI_GLOBAL_VARIABLE;
    CHAR16 *ptr;
    CHAR8 *str;
    UINT16 Handle;
    UINTN Index;
    UINT16 Length;

The information we want is not located in the firmware NVRAM as EFI variables. Instead we have to retrieve the information via SMBIOS.

The utility works by locating the base address of the SMBIOS table from the firmware EFI Configuration Table. The utility then walks down the SMBIOS table structures looking at the type of each structure. When it find a Type 0 (BIOS Information Indicator – designated in the SMBIOS standard to contain BIOS-related information) structure, it prints out the first three strings in that structure. The meaning of these three strings is specified by the SMBIOS standard, i.e. the first string is the BIOS Vendor’s Name, the second string is freeform and contains the BIOS Version and the third string in the structure is the BIOS Release Date in either *mm/dd/yy* or *mm/dd/yyyy* format. See Section 7.1 of the SMBIOS specification for more information.

[Updated 10/06/2012] Here is what is outputted by the utility on a Lenovo T430 laptop:
To compile this code, you must have the gnu_efi package installed. It is available on Fedora and most other Linux distributions. Due to problems in the definition of an SMBIOS structure in /usr/include/efi/libsmbios.h, comment out the libsmbios.h include directive in /usr/include/efi/efilib.h and use the EDK2 SmBios.h header instead. I have placed a copy of this header along with a Makefile and the above source code on GitHub under the showfirmware subdirectory.

By the way, do not confuse the EFI Configuration Table with the EFI System Table which is probably the most important data structure in EFI. The EFI Configuration Table is a member of the EFI System Table. A pointer to the EFI System Table is passed into each driver and application as part of its entry-point handoff. From the EFI System Table, an image can gain access to system configuration information (including the address of the SMBIOS table), EFI Boot Services, EFI Runtime Services, various Protocol services and more. You can get the Firmware Vendor and Firmware Revision information from the EFI System Table but not the Firmware Date.

See the DMTF SMBIOS and the UEFI specifications for more information.