American Megatrends

AMI Software Utility User Guide

Aptio AFU User Guide

Document Revision 1.11

Apr 28, 2017

NDA Required (NDA)

Legal

Table of Contents

Document Information

Purpose

This document provides information to use the Aptio AFU for updating system BIOS.

Audience

Generic BIOS Engineers, OEM Engineers, and Aptio Customers.

Change History

Date	Revision	Description
2013-11-19	1.00	Initial document created and update content to latest released of Afu
2014-03-20	1.01	Modified 0x18、0xB6、0xB7、0xBF、0xD0 error message text.
2014-04-24	1.02	Added error message 0x34、0x35.
2014-08-22	1.03	Added new commands for /meul and /JBC. Need to be updated the SmiFlash module to 5.001_SmiFlash_13.
2014-11-11	1.04	Added Windows 2012 R2 in the support list.
2015-01-30	1.05	Added new's command /cmd:.
2015-06-22	1.06	Removed Microsoft®DOS support.
2015-11-18	1.07	Added Linux Xen note.
2016-02-02	1.08	Added 0x4A、0x4B error message text. Added DOS does not support note.
2016-03-25	1.09	Added an announcement: Linux does not support Secure Boot
2016-10-18	1.10	Added support Linux Secure Boot. Added /CLRCFG, /BCPALL, /DPC command. Added 0x36, 0x37, 0x4C, 0x71 error messages.
2017-04-28	1.11	Added 0x4C, ROMLayout change error messages. Added answer for Windows digitally signed driver.

Introduction

Overview

AFU (AMI Firmware Update) is a package of utilities used to update the system BIOS under various operating systems. AFU only works for APTIO with SMI FLASH support.

AFU APTIO Features

This list of features is supported by command line, command prompt, EFI Shell, or BSD/Linux shell.

- Read system ROM image
- Flash ROM image
- Command line operating

Requirements

Supported Operating System

AFU is supported by the following operating systems:

- Microsoft® Windows® 2000
- Microsoft® Windows® XP
- Microsoft® Windows® 2003
- Microsoft® Windows® Server 2008 R2
- Microsoft® Windows® Server 2012 R2
- Microsoft® Windows® Vista
- Microsoft® Windows® 7
- Microsoft® Windows® 8
- Microsoft® Windows® 8.1
- Microsoft® Windows® 10
- Microsoft® Windows® PE
- EFI Shell Environment
- BSD
- Linux(*1)
- MS-DOS(*2)

Note:

*1. On Linux Xen environment, AFULNX must be executed in host desktop (Domain 0) of the virtual machine.

*2. DOS version is stopped supporting in AFU 3.08 or later version.

Firmware Requirements

- Compatible with Aptio 3, 4, and 4.5.
- Requires that the current installed firmware has SMI flashing support enabled.
- For supporting Secure Flash, the following eModules are required:

  - Secure Flash Pkg (4.6.5.1_SECMOD_003 or later)

  - CryptoPkg (4.6.5_CRYPTOAPI_0003 or later)

  - Capsule (4.5.6_Capsule_00 or later)

  - SMIFlash (4.6.3.6_SMIFLASH_23 or later)

  - OFBD (4.6.3.2_OFBD_1.0.2 or later)

  - OFBD Secure Flash (4.6.5.0_OFBD_SECURE_FLASH_0.0.5 or later)

Installation

To run, extract all of the files from the folder with the name corresponding to the desired operating system.

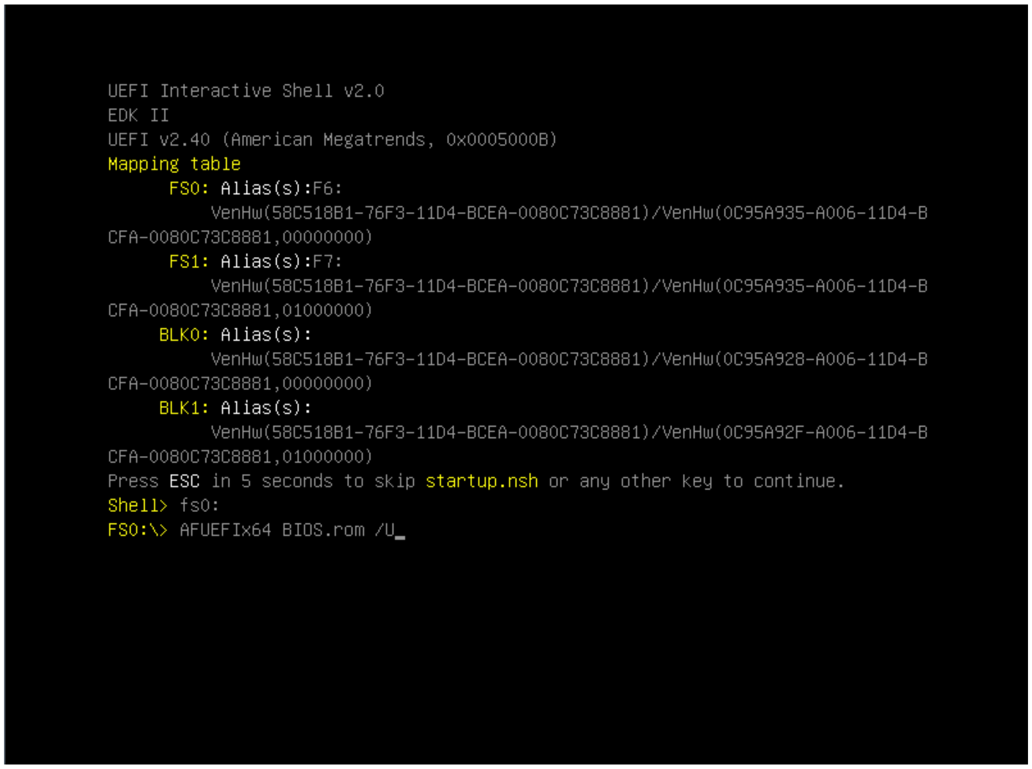
AFUAPTIO Operation

Overview

This mostly involves documenting all the SDL tokens and eLinks. This chapter explains the operation of AFUAPTIO.

The AFUAPTIO operation mode includes all of the AFUAPTIO features such as saving current ROM image to file, getting and displaying ROM ID from BIOS ROM file.

An example of AFUEFIX64 that getting and displaying ROM ID from BIOS ROM file command screen are shown below:



```
UEFI Interactive Shell v2.0
EDK II
UEFI v2.40 (American Megatrends, 0x0005000B)
Mapping table
      FS0: Alias(s):F6:
          VenHw(58C518B1-76F3-11D4-BCEA-0080C73C8881)/VenHw(0C95A935-A006-11D4-B
CFA-0080C73C8881,00000000)
      FS1: Alias(s):F7:
          VenHw(58C518B1-76F3-11D4-BCEA-0080C73C8881)/VenHw(0C95A935-A006-11D4-B
CFA-0080C73C8881,01000000)
     BLK0: Alias(s):
          VenHw(58C518B1-76F3-11D4-BCEA-0080C73C8881)/VenHw(0C95A928-A006-11D4-B
CFA-0080C73C8881,00000000)
     BLK1: Alias(s):
          VenHw(58C518B1-76F3-11D4-BCEA-0080C73C8881)/VenHw(0C95A92F-A006-11D4-B
CFA-0080C73C8881,01000000)
Press ESC in 5 seconds to skip startup.nsh or any other key to continue.
Shell> fs0:
FS0:\> AFUEFIx64 BIOS.rom /U_
```

Commands and Options

The following list is to offer you an overview of the commands and options provided by AFUAPTIO. The content can also be found in AFUAPTIO's help information. A more detailed usage of the commands and options will be explained in the next chapter.

Usage

AfuEfix64 <BIOS ROM File Name> [Option 1] [Option 2] …

Or

AfuEfix64 < Input or Output File Name > <Command>

Or

AfuEfix64 <Command>

BIOS ROM File Name

The mandatory field is used to specify path/filename of the BIOS ROM file with extension.

Commands

The mandatory field is used to select an operation mode.

- /O	Save current ROM image to file
- /U	Get and display ROM ID from BIOS ROM file
- /S	Refer to Option: /S
- /D	Verification test of given ROM File without flashing BIOS.
- /A	Refer to Option: /A
- /OAD	Refer to Option: /OAD
- /CLNEVNLOG	Refer to Option: /CLNEVNLOG

Options

The optional field is used to supply more information for flashing BIOS ROM. Following lists the supported optional parameters and format:

- /CLRCFG	Program without preserving setup configuration
-/BCPALL	Save all question values before flash
-/DPC	Don't Check Aptio 4 and Aptio 5 platform.
- /MEUL:	Program ME Entire Firmware Block, which supports Production.BIN and PreProduction.BIN files.
- /Q	Silent execution
- /X	Do not check ROM ID
- /CAF	Compare ROM file's data with Systems is different or not, if not then cancel related update.
- /S	Display current system's ROMID
-/JBC	Don't Check AC adapter and battery.
- /HOLEOUT:	Save specific ROM Hole according to given RomHole GUID.
- /SP	Preserve Setup setting.
- /R	Preserve all SMBIOS structures during programming.
- /Rn	Preserve SMBIOS type N during programming.(n=0-255)
- /B	Program Boot Block
- /P	Program main bios image
- /N	Program NVRAM
- /K	Program all non-critical blocks
- /Kn	Program n'th non-critical block (n=0-15)
- /HOLE:	Update specific ROM Hole according to RomHole GUID.
- /L	Program all ROM Holes
- /Ln	Program n'th ROM Hole only (n=0-15)
- /ECUF	Update EC BIOS when newer version is detected.
- /E	Program Embedded Controller block
- /ME	Program ME Entire Firmware Block.
- /MEUF	Program ME Ignition Firmware Block.

- /A	OEM Activation file.
- /OAD	Delete OEM Activation Key
- /CLNEVNLOG	Clear Event Log.
- /CAPSULE	Override Secure Flash policy by Capsule
- /RECOVERY	Override Secure Flash policy by Recovery
- /EC	Program Embedded Controller Block. (Flash Type)
-/CMD:	Send special command to BIOS. /CMD:{xxx}
- /REBOOT	Reboot after programming.
- /SHUTDOWN	Shutdown after programming.
- /FDR	Flash Flash-Descriptor Region.(*1)
- /GBER	Flash GBE Region.(*1)
- /MER	Flash Entire ME Region.(*1)
- /OPR	Flash Operation Region of SPS.(*1)
- /PDR	Flash PDR Region.(*1)

Note:

*1: If BIOS ME Module reports these commands, AFU will show this command.

To use a command of generic AFU on the Specific platform, please refer the help menu (/?) in generic AFU.

Rules

- Any parameter enclosed by < > is a mandatory field.
- Any parameter enclosed by [ ] is an optional field.
- <Commands> cannot co-exist with any [Options]. They are /O, /U, /D.
- Main BIOS image is default flashing area if no any options present.
- [/REBOOT], [/X], and [/S] will enable [/P] function automatically.
- If [/B] present alone, there is only the Boot Block area to be updated.
- If [/N] present alone, there is only the NVRAM area to be updated.
- If [/E] present alone, there is only the Embedded Controller block to be updated.

Usage

Overview

The AFUAPTIO offers the following basic command and option usages:

- AfuEfix64 <Input or Output File Name> [Option 1] [Option 2] …
- AfuEfix64 <Input or Output File Name> <Command>
- AfuEfix64 <Command>

Other usages which are not mentioned in help are:

- AfuEfix64 <ROM Hole File Name> <ROM Hole Option>:<ROM Hole GUID>
- AfuEfix64 <BIOS ROM File Name> <Option><Number>
- AfuEfix64 <Option /A> <OEM Activation Key Bin File Name>

These usages are explained in more detail in this chapter.

AfuEfix64 <Input or Output File Name> [Option 1] [Option 2] …

The user could put no option or combine multiple options in one command line. Commands cannot be combined in command line like options unless the command is categorized as both a command and an option, such as /S and /A.

For option combination case, AFUAPTIO will check its option priority list and execute the options according to the priority order. Three examples of this usage are provided below.

AfuEfix64 < BIOS ROM File Name>

Where BIOS ROM File Name, the mandatory field is used to specify path/filename of the BIOS ROM file with extension. This command line would trigger AFUAPTIO to run the default setting which flashes the system Main Block with the specified BIOS ROM File.

AfuEfix64 <Output BIOS ROM File Name> /D /S

Where Output BIOS ROM File Name, the mandatory field is used to specify path/filename of the BIOS ROM file with extension. /D is to verify the current BIOS and the BIOS ROM File, and /S, which is categorized as a command and also an option, gets and displays the current system's ROM ID.

AfuEfix64 <Output BIOS ROM File Name> /P /B /N /REBOOT

Where BIOS ROM File Name, the mandatory field is used to specify path/filename of the BIOS ROM file with extension. This command line is to flashing current BIOS by BIOS ROM file. /P /B /N are to specify that the flashing regions are Main Block, Boot Block and NVRAM. /REBOOT is to specify that reboot action will be performed in this execution. AFUAPTIO would execute the options in the order of /B, /P, /N and then reboot the system at the end. The order of execution is determined by AFUAPTIO design.

AfuEfix64 <ME File Name> /ME

Where ME File Name is used to specifying path/filename of the ME file with extension. This command line programs entire ME block with the specified ME file.

AfuEfix64 <Input or Output File Name> <Command>

AFUAPTIO can only execute one command at a time and it does not accept combinations of command and option in one command line except those can be both command and option. Three examples of this usage are provided below.

AfuEfix64 <Output BIOS ROM File Name> /O

Where BIOS ROM File Name, the mandatory field is used to specify path/filename of the BIOS ROM file with extension. This command line saves the current ROM image to a file.

AfuEfix64 <Output BIOS ROM File Name> /U

Where BIOS ROM File Name is used to specify path/filename of the BIOS ROM file with extension. This command line gets and displays the ROM ID from the specified BIOS ROM file.

AfuEfix64 <Command>

This command usage is for some commands which do not require inputting any file to complete the execution. Usually this type of commands accesses the current BIOS only. An example of this usage is provided:

AfuEfix64 /S

This command line gets and displays the ROM ID of the current BIOS in system.

AfuEfix64 <ROM Hole File Name> <ROM Hole Option>:<ROM Hole GUID>

This command usage is for outputting or flashing a certain ROM hole. For example, the command line for outputting a certain ROM hole whose GUID is 01234567- 89ab- cdef- 0123- 456789abcdef is as following:

AfuEfix64 <Output ROM Hole File Name> /HOLEOUT:0123456789abcdef0123456789abcdef

Where Output ROM Hole File Name is used to specify path/filename of the output ROM hole file with extension. The GUID after the option should not contain dashes or spaces in between.

Another example of flashing a certain ROM Hole whose GUID is 01234567- 89ab- cdef- 0123- 456789abcdef is as following:

AfuEfix64 <ROM Hole File Name> /HOLE:0123456789abcdef0123456789abcdef

Where ROM Hole File Name is used to specify path/filename of the ROM hole file with extension. Please discard dashes and spaces inside GUID line while typing.

AfuEfix64 <BIOS ROM File Name> <Option><Number>

This command usage is for /Kn and /Ln commands where n is indicating the numeric order of a certain non-critical block or ROM hole. For example, to program the 4th ROM hole, the command line could be:

AfuEfix64 <BIOS ROM File Name> /L4

Where BIOS ROM File Name is used to specify path/filename of the BIOS ROM file with extension, and 4 is to specify that the 4th ROM hole is the one to perform /L operation.

The next chapter has more detail of the numbering rule of non-critical blocks and ROM holes.

AfuEfix64 <Option /A> <OEM Activation Key Bin File Name>

This command usage is for /A command which insert a specific OEM activation key into the empty key inside current system BIOS. The command line is as follows:

AfuEfix64 /A <OEM Activation Key Bin File Name>

Where OEM Activation Key Bin File Name is used to specify path/filename of the OEM activation key file with extension. Please make sure that the OEM Activation Key region is empty before inserting the key, or please perform /OAD command before insertion.

Remarks

Overview

This chapter is to describe commands/options which require extra attention and to explain cases which may occur in certain unique scenarios.

Preserving Setup Setting – /SP

/SP command is designed specifically for "OEM NVRAM/Setup Variable Preserve" module part of OFBD. If /SP is called, AFUAPTIO would send SMI 0x26 twice to save setup setting before starting updating NVRAM and to restore setup setting after finishing updating NVRAM. Customer can customize their OFBD module to preserve certain NVRAM data when AFUAPTIO flashes the NVRAM area. For example, there are two methods for preserving Setup Password:

Method 1

Enable PRESERVE_PASSWORDS token – The BIOS will preserve its Setup password when AFUAPTIO calls the SMIFlash module.

Method 2

Control through /SP command – Customer can port PreserveSetupPassword in OFBDSETUPStoreHandle and RestoreSetupPassword in OFBDSETUPRestoreHandle, and use /SP command to keep or not to keep the Setup Password while updating the NVRAM:

```
AfuEfix64  xxx.ROM /N /SP - keep Setup password
AfuEfix64  xxx.ROM /N       - don't keep Setup password.
```

This feature needs BIOS' cooperation. To learn more about preserving setup data, please consult with your BIOS provider.

Preserving SMBIOS – /R and /Rn

If the SMBIOS data is stored in Main Block or Boot Block, AFUAPTIO /R and /Rn options would take the responsibility to preserve the SMBIOS data. If the SMBIOS data is stored in NVRAM and BIOS project's token SMBIOS_PRESERVE_NVRAM=0, the preservation process would take place at OFBD module. To know more about the detail of preserved data, please consult with your BIOS provider.

/R is used to preserve the whole SMBIOS data. To preserve a certain type of SMBIOS, please use /Rn. For example, to preserve SMBIOS Type 2 and Type 41 during BIOS flashing and the SMBIOS data is located in Boot Block, the command could be:

AfuEfix64 <BIOS ROM File Name> /B /R2 /R41

Programming NVRAM Region – /N

Erasing NVRAM may cause important variables lose.

Programming Specific NCB Block – /Kn

/Kn command is designed to program a specific non-critical block, or NCB block. AFUAPTIO would search ROM and identify the first NCB Block found as K0, and the second one as K1, etc. Therefore, command /K2 would program the third NCB Block found by AFU.

Programming Specific ROM Hole – /Ln

/Ln command is designed to program a specific ROM Hole. Each ROM Hole is identified in the following way: AFUAPTIO would search for ROM Holes in the order of Boot Block area and Main Block area, and identify each ROM Hole in consecutive integers from 0 to 15. So, for example, /L1 is used to program the second ROM Hole found in ROM.

Scenarios:

- If a ROM contains two ROM Holes in Boot Block area and two in Main Block area, AFUAPTIO would identify L0 and L1 for the two in Boot Block area and L2 and L3 for the two in Main Block area.

- If a ROM contains 2 ROM Holes in Boot Block area and none in Main Block area, AFUAPTIO would only find 2 ROM Holes in total and identify them as L0 and L1.

- If a ROM contains no ROM Holes in Boot Block area and three in Main Block area, AFUAPTIO would find nothing in Boot Block area and identify L0, L1 and L2 for the three ROM Holes in Main Block area.

Secured Flash Update – /CAPSULE and /RECOVERY

For Secured BIOS, the command rule for programming the current BIOS is different. There are two more modes, Capsule Mode and Recovery Mode, which are different from the regular Runtime Mode mentioned in the previous contents. Unlike Runtime Mode where all the commands/options are supported, Capsule Mode and Recovery Mode only support /P, /B, /N, and /E options, or depending on the BIOS design. The following description explains how to program BIOS under these two modes.

To override Secure Flash policy and program the BIOS image in Capsule Mode, please use the command:

AfuEfix64 <BIOS ROM File Name> /CAPSULE /P /B /N /E

And to override Secure Flash policy and program the BIOS image in Recovery Mode, please use this command:

AfuEfix64 <BIOS ROM File Name> /RECOVERY /P /B /N /E

Where BIOS ROM File Name is used to specify path/filename of the BIOS ROM file with extension.

For more detail on Secure Flash, please consult with your BIOS provider.

Send special command to BIOS – /CMD:{xxx}

Send the string between brackets to OFBD OEM CMD Checking Module. The string is corresponding to the string which is defined in BIOS by user.

Linux Pre-Requisites

1. Log in Linux as root otherwise use sudo (if permitted).

2. The compiler suite (gcc) must be installed. If these packages are not installed, the driver CANNOT be built.

3. For most of the distributions, AFU will generate driver without any notification, if it doesn't exist you need to install kernel sources. Also if Initmem fails, Please follow point 4.

4. Kernel sources must be installed, *CONFIGURED*, and then compiled. Following are steps to do this:

   a. Find Running Kernel's Configuration File:

   To configure the sources, simply change to the kernel source directory (typically **/lib/modules/$(uname -r)/build**). If it doesn't exist, you need to install kernel source.

   Typically, the reference configuration for the kernel can be found in the /boot directory with filename '**.config**', '**kernel.config**', or '**vmlinux-2.4.18-3.config**'. Type '**uname -a**' and use the configuration filename that best matches the output from '**uname -a**'. Also, check for **/dev/mem** directory existence. If it doesn't exist, you need to install kernel sources.

   Normally it comes with the installation unless if the option is deselected.

   On some distributions Red Hat for instance, there is a config directory under **/lib/modules/$(uname -r)/build**.

   Copy this configuration file into the root of the Linux kernel source tree (usually it is **/lib/modules/$(uname -r)/build)**. This file must be renamed to "**.config**"(dot config).

   b. Make Your AMI Flash Driver (**amifldrv_mod.o**):

   For most distribution, the command to build the driver is:

   ```
   afulnx_32 /MAKEDRV
                Or
   afulnx_64 /MAKEDRV
   ```

If your Linux's kernel source tree is under **/lib/modules/$(uname -r)/build**, instead of being in the default path '**/lib/modules/$(uname -r)/build**', then add a KERNEL flag:

```
afulnx_32 /MAKEDRV KERNEL=/lib/modules/$(uname -r)/build
                              Or
afulnx_64 /MAKEDRV KERNEL=/lib/modules/$(uname -r)/build
```

If KERNEL is omitted, the default path is /lib/modules/$(uname -r)/build.
This should work for MOST distributions.

c. Make Your AMI Flash Driver from driver source files (amifldrv_mod.o):

Using command /GENDRV, it will generate driver source files to specific directory.

```
afulnx_32 /GENDRV [Option 1] [Option 2]
                Or
afulnx_64 /GENDRV [Option 1] [Option 2]
```

Where,
[Option 1]: Specific kernel source 'KERNEL=XXXX' same as the /MAKEDRV
[Option 2]: Specific output directory 'OUTPUT=XXXX'

Generate files as outlined below:

```
File Name Description
-----------------------------------------------------------------------------
amiwrap.c Driver source code.
amiwrap.h Driver header.
amifldrv.o_shipped Object file for driver.
Makefile Makefile
-----------------------------------------------------------------------------
```

For most distribution, the command to build the driver is: make.

If your Linux's kernel source tree is under **/lib/modules/$(uname -r)/build**, instead of being in the default path '**/lib/modules/$(uname -r)/build**', then add a KERNEL flag:

```
make KERNEL=/lib/modules/$(uname -r)/build
```

If KERNEL is omitted, the default is **/lib/modules/$(uname -r)/build.**
This should work for MOST distributions.

d. Check Your Build:
Check the version of running Linux kernel with '**uname -r**'.
Check the version of **amifldrv_mod.o** with '**modinfo amifdrv_mod.o**'.
If they mismatch, you will need to select the correct configuration
File (.config), rebuild your kernel, and then rebuild your driver as described in steps a, b, c and d.

Signing Driver and Enrolling Public Key to the System

The following prerequisites are needed on the build system to sign the driver:

1. Login to Linux OS as root otherwise use sudo.
2. The compiler suite (gcc) must be installed. If it's not installed, the AFU driver cannot be built.
3. OpenSSL: Needed to generate cryptographic keys. OpenSSL tool can be downloaded from https://www.openssl.org
4. Perl interpreter: Needed to run the signing script. Perl tool can be downloaded from https://www.perl.org

Follow the below steps to sign the driver:

1. Boot to Linux OS.
2. Generate a Public and Private key pair using below openssl command: > openssl req -x509 -new -nodes -utf8 -sha256 -days 36500 -batch -configconfiguration_file.config -outform DER -out public_key.der -keyout private_key.priv

Note: The configuration file configuration_file.config must be created with the required information before running the command. A sample configuration file is shown below. The values in <> must be filled with actual values.

configuration_file.config:

```
[ req ]
default_bits = 4096
distinguished_name = req_distinguished_name
prompt = no
string_mask = utf8only
x509_extensions = myexts

[ req_distinguished_name ]
O = <organization_name>
CN = <organization_name> Signing Key
emailAddress = <email_address>

[ myexts ]
basicConstraints=critical,CA:FALSE
keyUsage=digitalSignature
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid
```

3. Build AFU driver using below command. The driver will be generated in the current directory with name amifldrv_mod.o.
```
> afulnx_64 /MAKEDRV
```

4. Execute below command to sign driver with the key generated in step 2.
```
> perl /usr/src/kernels/$(uname -r)/scripts/sign-file sha256 private_key.priv
public_key.der amifldrv_mod.o
```

5. Request addition of public key to MOK list using mokutil. The command will prompt a password which will be needed during public key enrollment in next step.
```
> mokutil --import public_key.der
```

6. Reboot the system which will launch MOK manager application to complete public key enrollment.

7. Once the public key enrollment is done, Boot to OS and execute below command to ensure the newly added key is available in system key ring.
```
> keyctl list %:.system_keyring
```

8. Install signed driver using insmod command.
```
> insmod amifldrv_mod.o
```

9. Ensure it is loaded successfully using lsmod command.

Reference: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/System_Administrators_Guide/sect-signing-kernel-modules-for-secure-boot.html

Support Table

Command/Option Support in Each Mode

Command	Runtime Mode	Capsule Mode	Recovery Mode
/O	Supported	Not Supported	Not Supported
/U	Supported	Not Supported	Not Supported
/S	Supported	Not Supported	Not Supported
/D	Supported	Not Supported	Not Supported
/A	Supported	Not Supported	Not Supported
/OAD	Supported	Not Supported	Not Supported
/CLNEVNLOG	Supported	Not Supported	Not Supported

Option	Runtime Mode	Capsule Mode	Recovery Mode
/MEUL	Supported	Not Supported	Not Supported
/Q	Supported	Not Supported	Not Supported
/X	Supported	Not Supported	Not Supported
/CAF	Supported	Not Supported	Not Supported
/S	Supported	Not Supported	Not Supported
/JBC	Supported	Not Supported	Not Supported
/SP	Supported	Not Supported	Not Supported
/R	Supported	Supported (*1)	Not Supported
/Rn	Supported	Supported (*1)	Not Supported
/B	Supported	Supported	Supported
/P	Supported	Supported	Supported
/N	Supported	Supported	Supported
/K	Supported	Not Supported	Not Supported
/Kn	Supported	Not Supported	Not Supported
/HOLE:	Supported	Not Supported	Not Supported
/HOLEOUT:	Supported	Not Supported	Not Supported
/L	Supported	Not Supported	Not Supported
/Ln	Supported	Not Supported	Not Supported
/ECUF	Supported	Not Supported	Not Supported
/E	Supported	Supported	Supported
/ME	Supported	Not Supported	Not Supported
/MEUF	Supported	Not Supported	Not Supported
/A	Supported	Not Supported	Not Supported
/OAD	Supported	Not Supported	Not Supported

Option	Runtime Mode	Capsule Mode	Recovery Mode
/CLNEVNLOG	Supported	Not Supported	Not Supported
/EC	Supported	Not Supported	Not Supported
/REBOOT	Supported	Not Supported	Not Supported
/SHUTDOWN	Supported	Not Supported	Not Supported

Note:

*** 1: This option must use with either /P or /B in order to be supported under Capsule Mode.**

Error Code Definition

CODE	Definition
0x01	Error: Unknown command.
0x02	Error: BIOS has no flash information available.
0x03	Error: ROM file size does not match existing BIOS size.
0x04	Error: ROM file ROMID is not compatible with existing BIOS ROMID.
0x05	Error: Bootblock error.
0x06	Error: This BIOS version has more Non-Critical blocks than supported.
0x07	Error: BIOS checksum error.
0x08	Error: Invalid option
0x09	Error: Size of ROM file does not match the size of system ROM
0x0A	Error: Unable to update ROM hole
0x0B	Error: ROMHOLE not exist
0x0C	Error: BIOS update cancelled by user.
0x0D	Error: BIOS Report Error.
0x0E	Error: Kernel source files cannot be found.
0X0F	Error: Size of PLDM file is more than the FV size.
0x10	Error: Unable to load driver.
0x11	Error: Unable to unload driver.
0x12	Error: No non-critical blocks found in ROM file.
0x13	Error: Requested non-critical block not available in ROM file.
0x14	Error: Non-critical blocks in ROM image file do not match those in the system.
0x15	Error: Secure Flash function is not supported on this platform.
0x16	Error: Unable to get Secure Flash policy from BIOS.
0x17	Error: Unsupported Secure Flash policy.
0x18	Error: Secure Flash Rom Verify fail.
0x19	Error: Failed to erase flash chip (at Runtime Secure Flash).
0x1A	Error: Failed to update flash chip (at Runtime Secure Flash).
0x1B	Error: Failed to read flash chip (at Runtime Secure Flash).
0x1C	Error: Failed to verify flash chip (at Runtime Secure Flash).
0x1D	Error: Failed to load image into memory.
0x1E	Error: Secure Flash function is not supported on this file.
0x1F	Error: Reserved for Secure Flash.
0x20	Error: Unable to initialize memory manager.
0x21	Error: Unable to close memory manager.

0x22	Error: Problem allocating memory.
0x23	Error: Problem freeing memory.
0x24	Error: Problem allocating BIOS buffer.
0x25	Error: Problem freeing BIOS buffer.
0x26	Error: Problem freeing mapping BIOS.
0x27	Error: Problem freeing unmapping BIOS.
0x28	Error: Problem mapping BIOS data.
0x29	Error: Problem unmapping BIOS data.
0x30	Error: Problem opening file for reading.
0x31	Error: Problem reading file.
0x32	Error: Problem opening file to write.
0x33	Error: Problem writing file.
0x34	Error: Using the wrong AFU version, Please use Aptio 4 AFU.
0x35	Error: Using the wrong AFU version, Please use Aptio 5 AFU.
0x36	Error: Fail with problem of ESP Driver init.
0x37	Error: Fail with problem of copy ROM file to ESP driver.
0x40	Error: BIOS is write-protected.
0x41	Error: Can not close flash interface.
0x42	Error: Problem reading flash.
0x43	Error: Problem erasing flash.
0x44	Error: Problem writing flash.
0x45	Error: Problem verifying flash.
0x46	Error: Problem getting flash information.
0x47	Error: No firmware id.
0x48	Error: Power cord not connected. Plug in power cord to flash.
0x49	Error: A platform condition has prevented flashing.
0x4A	Error: Platform data is not empty, And data address is not Alignment Block Address.
0x4B	Error: SLP key is not empty at all.
0x4C	Error: Rom file ROM layout is changed.
0x50	Error: This program must be run in MS-DOS mode.
0x60	Error: Accessing registry.
0x61	Error: Program already running.
0x70	Error: BSD access IO.
0x71	Error: Linux does not support Auto Build Driver when Secure Boot Enable.
0x80	Error: Size of system ROM mismatches size of ROM file
0x81	Error: ROM ID mismatch
0x82	Error: Bootblock checksum error
0x90	Error: Error to shutdown
0x91	Error: Error to restart...
0x92	Error: Can't open ROM ID file
0x93	Error: ROM ID file is not a ROM file.
0x94	Error: Invalid MAC address
0x95	Error: Invalid load current CMOS option
0x96	Error: Invalid retry count
0x97	Error: Invalid defined ROM ID length
0x98	Error: Invalid SMI

0x99	Error: ROM File ID don't exist
0x9A	Error: System ROM ID don't exist
0x9B	Error: Password Retry count exceeded.
0x9C	Error: BIOS don't support NVRAM/SETUP preserve function
0x9D	Error: Store SETUP setting error
0x9E	Error: Restore SETUP setting error
0x9F	Error: Cannot analyze ROM file. ROM file may be corrupted
0xA0	Error: Cannot analyze the ME Data. ROM file may be corrupted
0xA1	Error: BIOS does not support ME Entire Firmware update
0xA2	Error: BIOS does not support ME Ignition Firmware update
0xA3	Error: Invalid EC ROM file
0xA4	Error: EC ROM file checksum error
0xA5	Error: Can't enter EC flash mode
0xA6	Error: Erasing EC flash memory fail
0xA7	Error: Initial EC programming fail
0xA8	Error: EC flash data transmit error
0xA9	Error: Writing EC flash memory fail
0xAA	Error: Exit EC programming mode fail
0xAB	Error: ROM Chip ID mismatch
0xAC	Error: Invalid EC Header Table
0xAD	Error: EC does not permit BIOS update
0xAE	Error: BIOS doesn’t support OEMCMD function
0xAF	Error: Store DMI Data error
0xB0	Error: Restore DMI Data error
0xB1	Error: Invalid Activation Key file.
0xB2	Error: File Size is greater than image activation key length.
0xB3	Error: Image activation key larger than BIOS activation key.
0xB4	Error: Activation Key checksum error.
0xB5	Error: No Support Activation Key error.
0xB6	Error: OA key is available, and OA Key is not the same as BIN file in the system.
0xB7	Error: OA key is empty.
0xB8	Error: OA key region incorrect.
0xB9	Error: BIOS doesn’t support Clear event log function.
0xBA	Error: Clear event log error.
0xBB	Error: Rom image layout detected RomHole is redesigned.
0xBC	Error: BIOS have more than one RomHole’s GUID is the same.
0xBD	Error: Requested Rom Hole not available in ROM file.
0xBE	Error: RomHoles in ROM image file do not match those in the system.
0xBF	Error: OA key is available, and OA Key is the same as BIN file in the system.
0xC0	Error: BIOS doesn’t support process ME information
0xC1	Error: BIOS return error, when trying to re-flash ME Firmware data.
0xC2	Error: Region is write-protected
0xC6	Error: No EC blocks found in system ROM.
0xC7	Error: BIOS doesn’t support all ROM flashing function.
0xD0	Error: OA key data is invalid.
0xD1	Error: BIOS has already updated OA.

0xD2	Error: BIOS does not allow updating OA.
0xD3	Error: BIOS doesn’t support updating OA.
0xD4	Error: The DMI data size of system is greater than File's DMI data length.
0xD5	Error: BIOS doesn't support EC Battery Check function.

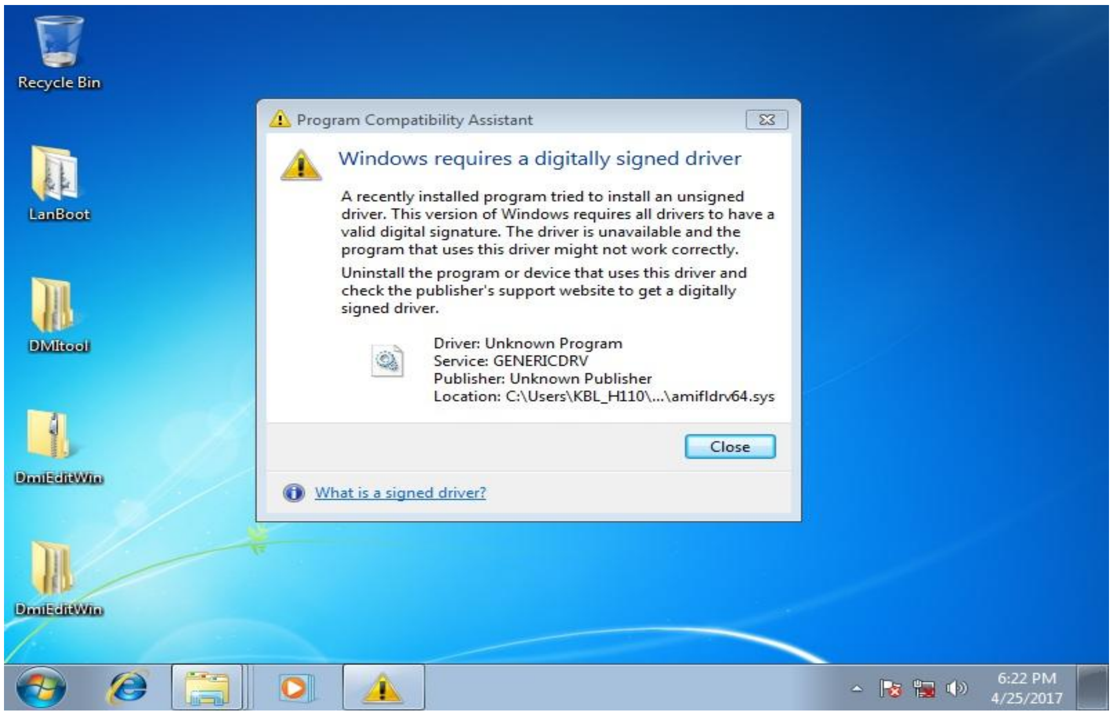
FQA

The Error Message Information of ROM

AFU has added the check mechanism of ROM information. AFU would compare the information between updated ROM and on board ROM. If these ROMs have different information, AFU will show the error of 0x4C.

AMI extremely suggest users to stop choosing “Continue to update” if users do not comprehend ROM structure very much. The system will be crashed after BIOS update because of ROM information difference.

Windows requires a digitally signed driver



Windows requires a digitally signed driver

A recently installed program tried to install an unsigned driver. This version of Windows requires all drivers to have a valid digital signature. The driver is unavailable and the program that uses this driver might not work correctly.

Uninstall the program or device that uses this driver and check the publisher's support website to get a digitally signed driver.

Driver: Unknown Program
Service: GENERICDRV
Publisher: Unknown Publisher
Location: C:\Users\KBL_H110\...\amifldrv64.sys

This issue is resolved by a security fix provided by MS. KB3033929 resolves this issue. The certificate used to sign the driver is higher security and older versions of Win7 don’t support it.