## AMI Software Utility User Guide

# Aptio 5.x AFU User Guide

**Document Revision 1.09**

**December 21, 2021**

Public Document
(PUB)

# Legal

Disclaimer

This publication contains proprietary information which is protected by copyright.  No part of this publication may be reproduced, transcribed, stored in a retrieval system, translated into any language or computer language, or transmitted in any form whatsoever without the prior written consent of the publisher, American Megatrends International LLC. American Megatrends International LLC retains the right to update, change, modify this publication at any time, without notice.

For Additional Information

Call American Megatrends International LLC. At 1-800-828-9264 for additional information.

Limitations of Liability

In no event shall American Megatrends be held liable for any loss, expenses, or damages of any kind whatsoever, whether direct, indirect, incidental, or consequential, arising from the design or use of this product or the support materials provided with the product.

Limited Warranty

No warranties are made, either expressed or implied, with regard to the contents of this work, its merchantability, or fitness for a particular use.  American Megatrends assumes no responsibility for errors and omissions or for the uses made of the material contained herein or reader decisions based on such use.

Trademark and Copyright Acknowledgments

Copyright © 2021 American Megatrends International LLC.  All Rights Reserved.

American Megatrends International LLC
5555 Oakbrook Parkway
Building 200
Norcross, GA 30093 (USA)

All product names used in this publication are for identification purposes only and are trademarks of their respective companies.

# Table of Contents

## Document Information

## Purpose

This document provides information to use the Aptio 5.x AFU for updating system BIOS.

## Audience

Generic BIOS Engineers, OEM Engineers, and Aptio Customers.

## Change History

| Date | Revision | Description |
|---|---|---|
| 2019-10-24 | 1.00 | This is the PUB version, this first draft is based on NDA version 1.26 to generate. |
| 2019-12-03 | 1.01 | Change the text description of RLC: E to "entire BIOS region". |
| 2020-01-15 | 1.02 | Add new command **/RLC:M**. |
| 2020-05-05 | 1.03 | Added 0x100 error message.<br>Added new command **/PFRU、/PFRA、/PFRD、/PFRR、/PFRC、/PFRW** |
| 2020-07-09 | 1.04 | Added Linux support 20.04<br>Added Linux driver description<br>Added RLC function description<br>Added 0x5C,0x5A,0xC9,0xCA error messages. |
| 2020-09-17 | 1.05 | Added FAQ for AFU Windows version for adding additional driver description. |
| 2020-12-09 | 1.06 | Added new command **/PFRC:CPLD**, **/K:<GUID>**. |
| 2021-04-16 | 1.07 | Added requirements for **make** and **libelf-dev** in Linux prerequisites. |
| 2021-10-26 | 1.08 | Added AFU's behavior pattern when BIOS Guard is enabled.<br>Added 0x101, 0x102, 0x120, 0x121, 0x122, 0x123, 0x124, 0x125, 0x126, 0x127, 0x128 error messages. |
| 2021-12-21 | 1.09 | Added new command **/FAB**.<br>Update Linux signature driver reference link. |

# Introduction

## Overview

**A**FU (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.

## 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

## Notifications

- BIOS update may have some potential risks so that AMI suggests closing all programs users are processing and stopping the anti-virus software temporarily!
- Please DO NOT power off or restart the computer device when the system is reading BIOS or updating BIOS!
- To avoid failing in BIOS update, please DO NOT remove the hard disk or USB or any devices. When users update BIOS in any inappropriate way, that incorrect behavior will result in BIOS crash and the computer devices cannot be powered on.

## Requirements

### Supported Operating System

AFU is supported by the following operating systems:

- ~~Microsoft® Windows® Server 2012 R2~~
- ~~Microsoft® Windows® 7~~
- Microsoft® Windows® 8
- Microsoft® Windows® 8.1
- Microsoft® Windows® Server 2016
- Microsoft® Windows® Server 2019
- Microsoft® Windows® Server 2022

- Microsoft® Windows® 10
- Microsoft® Windows® 11
- Microsoft® Windows® PE
- EFI Shell Environment
- BSD
- Linux(*1)
  - ✓ *Ubuntu (20.04)*
  - ✓ *Red Hat*
  - ✓ *Fedora (29)*
  - ✓ *openSUSE*
  - ✓ *Debian (9.6)*
  - ✓ *CentOS (7.3)*

**Note:**

1. Linux notes:

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

   b. The version of Linux distribution listed which version that AMI testing.

2. Due to System IO access, Windows version requires administrator privileges and executes with "Run as Administrator" option.
3. Due to System IO access, Linux version requires root authority.
4. AFUWIN still can be run while using unsupported Microsoft Windows versions.

## Firmware Requirements

- Compatible with AptioV.

- Please contact AMI for more details.

# AFU Operation

## Overview

The AFU operation mode includes all of the AFU 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 AFU. The content can also be found in 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.

- /FV            Copy Secure FV to EFI(*2)
- /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     Clear Event Log

## Options

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

- /SSB:          Send String to BIOS. For example, /SSB:{xxx}(*3)

- /CMD:          Send special command to BIOS. /CMD:{xxx}

- /OEMCMD:       Send special value to BIOS. /OEMCMD:xxx

- /DPC           Don't Check Aptio 4 and Aptio 5 platform

- /PW:           Input password for file

- /MEUL:         Program ME Entire Firmware Block, which supports Production.BIN and PreProduction.BIN files

- /Q             Silent execution

- /X             Do not check ROM ID

- /ATR:          Select AMI Twins ROM to flash. For example, /ATR:D or ATR:U(*3)

- /ATR           Select Another Tank ROM to flash. (*3)

- /S             Display current system's ROMID

- /JBC           Don't Check AC adapter and battery

- /CLRCFG        Program without preserving setup configuration

- /BCPALL        Save all question values before flash

- /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

- /K:<GUID>      Program non-critical block by GUID.

- /Kn            Program n'th non-critical block (n=0-15)

- /RLC:          To set default option for Rom layout change (E: Entire BIOS region, A: Abort, F: Force, M:Mix Non-BIOS region command.)

- /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

- /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

- /RECOVERY:ESP   Override Secure Flash policy by ESP Partition Recovery

- /EC         Program Embedded Controller Block (Flash Type)

- /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)

- /PFRU      Updates platform firmware (*3)

- /PFRU:CPLD    Updates platform firmware (*3)

- /PFRA      Signal active image update intent  (*3)

- /PFRD      Signal dynamic region update of active image  (*3)

- /PFRR      Signal recovery image update intent  (*3)

- /PFRC      Signal CPLD update intent  (*3)

- /PFRW     Wait until reset to update  (*3)

- /FAB       Program PSP block  (*3)

**Note:**

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

*2: If AFU is running on Linux OS and Windows OS, AFU will show this command.

*3: If BIOS Module support, AFU will show this command.

To use a command of generic AFU on the Specific platform, please refer to 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.

- Main BIOS image is default flashing area if no options are present.

- [/REBOOT], [/X], and [/S] will enable [/P] function automatically.

- If [/B] stands alone, there is only the Boot Block area to be updated.

- If [/N] stands alone, there is only the NVRAM area to be updated.

- If [/E] stands alone, there is only the Embedded Controller block to be updated.

- All options are case-insensitive and have no order.

# Usage

## Overview

The AFU 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, AFU will check its option priority list and execute the options according to the priority order. Three examples of this usage are provided below.

### AfuEfix64 <Input 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 AFU to run the default setting which flashes the system Main Block with the specified BIOS ROM File, so the default behavior will be same as "AfuEfix64 <Input BIOS ROM File Name> /P".

### AfuEfix64 <Input 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 <Input 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. AFU 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 AFU design.

### AfuEfix64 <ME File Name> /ME

Where ME File Name is used for 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>

AFU 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 <Input 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 the 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 options 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 the 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 inserts 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.

# Use case

## 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 option is designed specifically for "OEM NVRAM/Setup Variable Preserve" module part of OFBD. If /SP is called, AFU 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 AFU 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 AFU 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 more cooperations from BIOS side. To learn more about preserving setup data, please consult with your BIOS provider.

**Note:**

**/SP** is not required when using the **_BIOS Configuration Preserve_** feature.

# Preserving SMBIOS – /R and /Rn

If the SMBIOS data is stored in Main Block or Boot Block, AFU /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 to lose. For the needs to preserve important variables, please check previous "Preserving Setup Setting - /SP" and later "Platform Level Data Model (PLDM)" sections.

# Programming Specific NCB Block – /Kn

/Kn command is designed to program a specific non-critical block or NCB block. AFU 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: AFU 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, AFU 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, AFU 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, AFU 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.

# Send special value to BIOS – /OEMCMD:xxx

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

# To set default option for ROM Layout Changed – /RLC:

When AFU detects ROM layout change, AFU will hold the flash step and wait for users to input which command they need:

"/RLC:E"  - This option will update entire BIOS region and exit.

"/RLC:A" – This option will be no ROM update.

"/RLC:F"  - This option will be forcing to follow the command by user provided.

"/RLC:M" – This option will update entire BIOS region and Mix Non-BIOS region command.

This command can input a default setting directly.

Normally, AFU will hold 5 seconds to be reconsidered by "continue" or "cancel". However, if users input the command /RLC: (E or A or F or M) and command /Q, AFU will not hold 5 seconds.

If AFU does not detect ROM layout change but users input this command, it will not be workable.

**Note:** AFUWINGUI does not support this command.

---

# Don't Check AC adapter and battery – /JBC

By default, AFU will perform power checking (AC adapter and battery checking). Users can use /JBC option to skip power checking. How AFU check power status?

| How AFU Check Power Status<br><br>(Power: AFU won't know the power is from AC or battery)<br><br>(Status: AFU won't know the charge level, only know status) | ⇨ | a. By checking value stored in Embedded Controller (EC) firmware | ⇨ Unable to find value, or No EC | ⇨ | **Continue** |
| | | | ⇨ Value said "Status Normal" | ⇨ | **Continue** |
| | | | ⇨ Value said "Status Abnormal" | ⇨ | **Won't Flash** |
| | ⇨ | b. Through OFBD module "AC/Battery Checking" module part | ⇨ No OFBD, or too old to support | ⇨ | **Continue** |
| | | | ⇨ OFBD returns "Status Normal" | ⇨ | **Continue** |
| | | | ⇨ OFBD returns "Status Abnormal" | ⇨ | **Won't Flash** |

There are 2 ways AFU check power status:

a. By checking value stored in Embedded Controller (EC) firmware:
By searching EC binary data start from 0xE000 and using "$ECB" signature to find power status value.

b. Through OFBD module "AC/Battery Checking" module part.

AFU will only perform (a) (b) for power checking. AFU will stop flash process if (a) or (b) return error. It will also show both error messages from (a) (b) and AFU error code "0x0D – BIOS Report Error."

AFU will keep executing without doing (a) (b) if no EC or no OFBD module "AC/Battery Checking" function.

"Does the tool exit if battery only?"

AFU won't know the power is from AC or battery. AFU will exit only when the return value of (a) or (b) is error. So AFU won't exit if "Battery only and return OK".

"Does it check the level of charge?"

AFU won't know the charge level, only know the power status "OK/Normal or Error/Abnormal" returned by (a) or (b).

# Linux Pre-Requisites

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

2. The compiler suite (gcc, make, libelf-dev) 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 a specific directory.

afulnx_32 /GENDRV [Option]
or
afulnx_64 /GENDRV [Option]

Where,
[Option]: Specific kernel source 'KERNEL=XXXX' same as the /MAKEDRV

Generate files as outlined below:

File Name Description
AFU5.11.06.1854  or previous versions
---------------------------------------------------------------------------
amiwrap.c Driver source code.
amiwrap.h Driver header.
amifldrv.o_shipped Object file for the driver.
Makefile Makefile
---------------------------------------------------------------------------
AFU5.12.00.1904 or later versions
---------------------------------------------------------------------------
amiwrap.c Driver source code.
amifldrv.h Driver header.
amifldrvdefs.h Driver struct define.
amifldrv.o_shipped Object file for the 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.

5. The case of Linux driver:

|  | Secure Boot Enabled | Secure Boot Disabled |
|---|---|---|
| **WSMT is supported** | Driver required | Driver no required |
| **Can access file path:/dev/mem** | Driver required | Driver no required |
| **Run Time Memory Hole support** | Driver required | Driver no required |

# 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 –config configuration_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=eyed
```

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/$(uname –r)/scripts/sign-file sha256 private_key.priv public_key.der amifldrv_mod.o
Or
> /usr/src/kernels/$(uname –r)/scripts/sign-file sha256 private_key.priv public_key.der amifldrv_mod.o

5. Request addition of a 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

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

6-1. Select Enroll MOK.
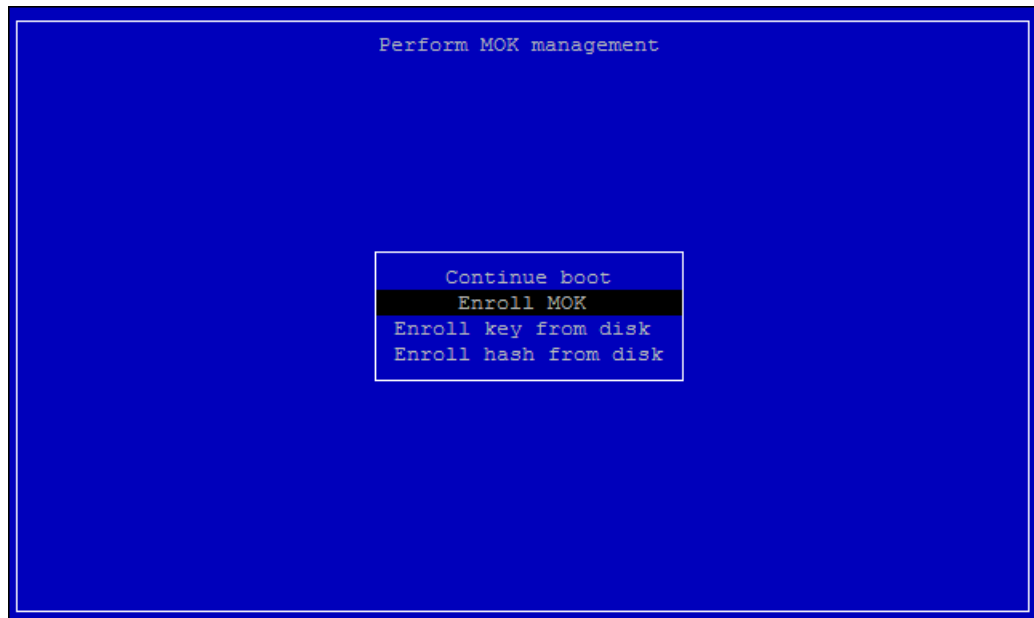


6-2. Select Continue.
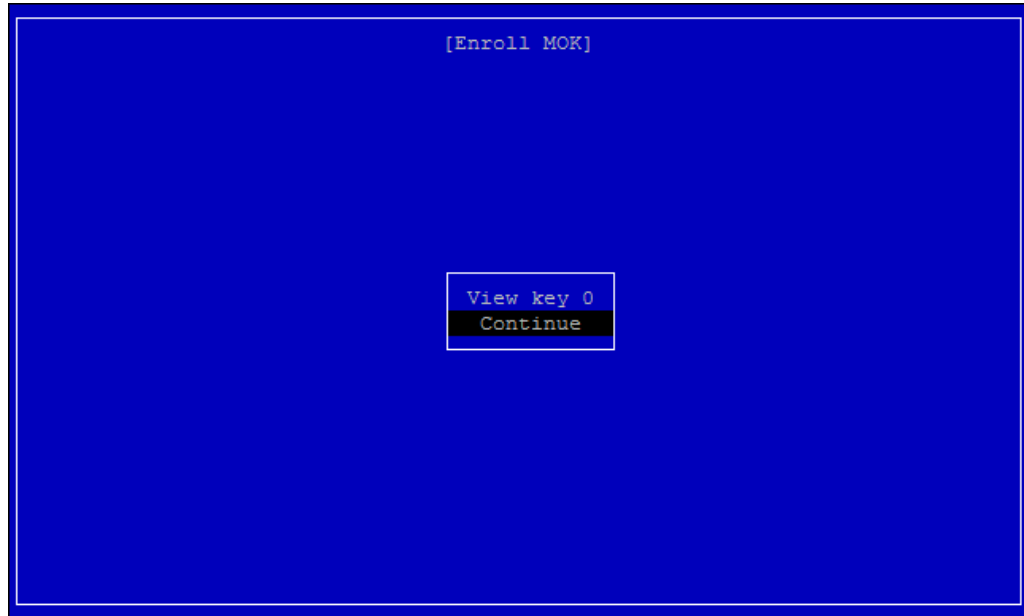
6-3. Select Yes.



6-4. Input step 5 password.
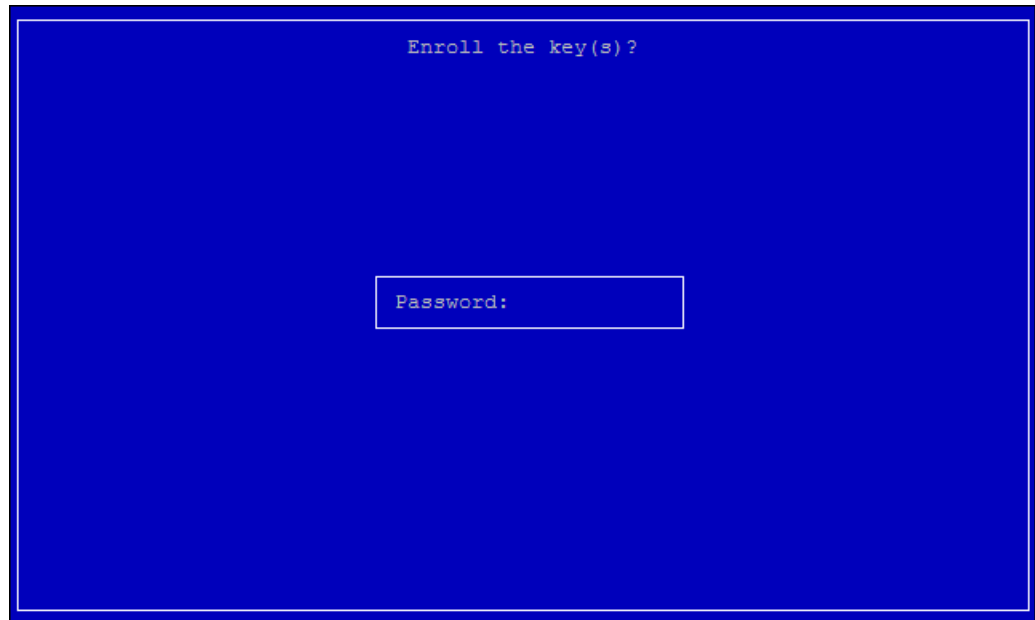
7. Once the public key enrollment is done, Boot to OS and execute below command to ensure the newly added key is available in system key ring.

> keyctl list %:.system_keyring

Or

> keyctl list %:.builtin_trusted_keys

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/8/html/managing_monitoring_and_updating_the_kernel/signing-kernel-modules-for-secure-boot_managing-monitoring-and-updating-the-kernel

# Intel BIOS Guard Support

## Overview

Intel BIOS Guard is a secure method of flashing, it must use the interface of Intel to flash, the original AFU that it flashed BIOS via the SMI flash module, when the BIOS Guard feature is enabled, it only could use the BIOS Guard interface to update SPI data which are the blocks guarded by BIOS Guard, while using the SMIFlash interface will be blocked.

The previous AFU only supported /Recovery and /Capsule commands. After 5.15.00, AFU can support BIOS Guard commands and AFU commands under BIOS Guard enabled. In this state, AFU will depend on the format of the input file to allow AFU for supporting AFU commands or BIOS Guard commands.

## File Format

The file size of BIOS Guard is larger than Aptio's, because it included the file header of BIOS Guard and the flash commands that can be used.

When the file is enabled or disabled by the BIOS Guard feature, the following symptoms will occur.

| File format | BIOS Guard enabled | BIOS Guard disabled |
|---|---|---|
| Aptio Image File | Error 0x126, Error 0x127, Error 0x128 | Normal execution |
| BIOS Guard File | Normal execution | Error 0x120 |

## Flash Behavior

Since the protection level of BIOS Guard is higher than Secure flash, the flash mode using SMI Flash interface will be blocked by BIOS Guard. When BIOS Guard function is enabled and users using Aptio V Image, AFU will use SMI Flash interface for flashing, but since BIOS Guard is enabled, most of the blocks will be blocked and show error 0x126.

When BIOS Guard is enabled, the process of using BIOS Guard ROM File in Flash will also start to do the following checks like using Aptio ROM Image.

1. EC Battery Check.

2. ROM ID Check.

3. ROM File Checksum.

4. OFBD Password check.

5. Support PLDM feature.

When BIOS Guard is enabled, AFU command support is added to the following sections when using the BIOS Guard ROM file.

| Command String | Description |
|---|---|
| /Q | Silent execution. |
| /X | Don't Check ROM ID. |
| /CLRCFG | Program without preserving setup configuration. |
| /BCPALL | Save all question values before flash. |
| /MLANG: | Multiple mapping language support. |
| /REBOOT | Reboot after programming. |
| /SHUTDOWN | Shutdown after programming. |

When BIOS Guard is enabled, some functions may not be supported when using Aptio ROM File for flashing.

| Featuer | Symptom |
|---|---|
| Capsule Update | Display error message Error 0x127. |
| Update Auto ME Firmware | Display error message Error 0xC1. |
| Update ME Firmware Block | Display error message Error 0x128, or stop updating. |

# AMD Combo AM4 Support

AFU has supported AMD Combo AM4 update on platform version, MyrtleQogir_1AVUD (5.14_VEB_1AVUD002 or later), since AFU 5.11.05.1840. This support will need a specific ROM by particular package to generate a 32 MB ROM file.

AFU offers command /ATR:U, /ATR:D to be chosen by users for which ROM region should be flashed. BIOS offers command /CMD:{TOP16M}, /CMD:{BOTTOM16M} to be chosen by users for which ROM region should be flashed.

| ATR:U | NVRAM | BOTTOM16M |
| | FV_MAIN | |
| | FV_BB | |
| ATR:D | NVRAM | TOP16M |
| | FB_MAIN | |
| | FV_BB | |

If users want to flash ROM of ATR:D to TOP16MB, the command will be /CMD:{TOP16M}

If users want to flash ROM of ATR:U to BOTTOM16MB, the command will be /CMD:{BOTTOM 16M}

If users want to know more details with regard to AMD Combo AM4 update information, please contact AMD BIOS Team.

# Support Table

## Command/Option Support in Each Mode

| Command | Runtime Mode | Capsule Mode | Capsule OACU Mode | Recovery Mode |
|---|---|---|---|---|
| /O | Supported | Not Supported | Not Supported | Not Supported |
| /U | Supported | Not Supported | Not Supported | Not Supported |
| /S | Supported | Not Supported | Not Supported | Not Supported |
| /D | Supported | Not Supported | Not Supported | Not Supported |
| /A: | Supported | Not Supported | Not Supported | Not Supported |
| /OAD | Supported | Not Supported | Not Supported | Not Supported |
| /CLNEVNLOG | Supported | Not Supported | Not Supported | Not Supported |

| Option | Runtime Mode | Capsule Mode | Capsule OACU Mode | Recovery Mode |
|---|---|---|---|---|
| /CMD: | Supported | Not Supported | Not Supported | Not Supported |
| /OEMCMD: | Supported | Not Supported | Not Supported | Not Supported |
| /DPC | Supported | Not Supported | Not Supported | Not Supported |
| /PW: | Supported | Not Supported | Not Supported | Not Supported |
| /MEUL: | Not Supported | Supported | Not Supported | Not Supported |
| /Q | Supported | Not Supported | Not Supported | Not Supported |
| /X | Supported | Not Supported | Supported | Not Supported |
| /S | Supported | Not Supported | Not Supported | Not Supported |
| /JBC | Supported | Not Supported | Not Supported | Not Supported |
| /CLRCFG | Supported | Not Supported | Supported | Not Supported |
| /BCPALL | Supported | Not Supported | Not Supported | Not Supported |
| /HOLEOUT: | Supported | Not Supported | Not Supported | Not Supported |
| /SP | Supported | Not Supported | Not Supported | Not Supported |
| /R | Supported | Supported ( *1 ) | Supported ( *1 ) | Not Supported |
| /Rn | Supported | Supported ( *1 ) | Supported ( *1 ) | Not Supported |
| /B | Supported | Supported | Supported | Supported |
| /P | Supported | Supported | Supported | Supported |
| /N | Supported | Supported | Supported | Supported |
| /K | Supported | Not Supported | Supported | Not Supported |
| /Kn | Supported | Not Supported | Supported | Not Supported |
| /RLC: | Supported | Not Supported | Not Supported | Not Supported |
| /HOLE: | Supported | Not Supported | Not Supported | Not Supported |
| /L | Supported | Not Supported | Supported | Not Supported |
| Option | Runtime Mode | Capsule Mode | Capsule OACU Mode | Recovery Mode |

| /Ln | Supported | Not Supported | Supported | Not Supported |
|------|-----------|---------------|-----------|---------------|
| /ECUF | Supported | Not Supported | Not Supported | Not Supported |
| /E | Supported | Supported | Supported | Supported |
| /ME | Supported | Not Supported | Not Supported | Not Supported |
| /A: | Supported | Not Supported | Not Supported | Not Supported |
| /OAD | Supported | Not Supported | Not Supported | Not Supported |
| /CLNEVNLOG | Supported | Not Supported | Not Supported | Not Supported |
| /CAPSULE | Supported | Supported | Supported | Not Supported |
| /RECOVERY | Supported | Not Supported | Not Supported | Supported |
| /EC | Supported | Not Supported | Not Supported | Not Supported |
| /REBOOT | Supported | Not Supported | Not Supported | Not Supported |
| /SHUTDOWN | Supported | Not Supported | Not Supported | Not Supported |

**Note:**

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

# Error Codes

## 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 canceled 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 the problem of ESP Driver init. |
| 0x37 | Error: Fail with the problem of copy ROM file to ESP driver. |
| 0x38 | Error: Multi tank ROMs' ROM ID are not allowed the same. |
| 0x39 | Error: Multi tank ROMs' ROM ID do not match platform ROM ID. |
| 0x3A | Error: BIOS does not support ACAU Module. |
| 0x3B | Error: AFU does not support this ACAU version. |
| 0x3C | Error: BIOS ACAU reports error. |
| 0x3D | Error: BIOS does not support ACAU-SSB function. |
| 0x3E | Error: BIOS ACAU-SSB reports error. |
| 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. |
| 0x4D | Error: Get block size error. |
| 0x50 | Error: This program must be run in MS-DOS mode. |
| 0x51 | Error: Size of PLDM file is more than the FV size. |
| 0x52 | Error: Preserving setup configuration failed. |
| 0x53 | Error: BIOS does not support preserving the setup configuration. Use /CLRCFG option to flash without preserving. |
| 0x54 | Error: Initialization failed for preserving setup configuration. |
| 0x55 | Error: Error occurred while retrieving HII data. |
| 0x56 | Error: Error occurred while creating PLDM Table. |
| 0x57 | Error: PLDM table data is empty. |
| 0x58 | Error: Capsule command can not be used alone, must be used with the /p /b /n option. |

| 0x59 | Error: Recovery command can not be used alone, must be used with the /p /b /n option. |
|------|---------------------------------------------------------------------------------------|
| 0x5A | Error: EMMC Mode cannot be used runtime flash command, please use capsule or recovery to update BIOS. |
| 0x5B | Error: Capsule and Recovery command cannot be used together. |
| 0x5C | Error: BIOS doesn't support SPS recovery function. |
| 0x5D | Error: BIOS doesn't support process SPS recovery information. |
| 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. |
| 0x83 | Error: Your BIOS policy does not allow ROMID check bypassing. |
| 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 doesn't exist. |
| 0x9A | Error: System ROM ID doesn'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 an 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. |
| 0xC8 | Error: Invalid ROM image file, ROM image file may be corrupted. |
| 0xC9 | Error: Invalid SPS ME Reocvery ROM image file, ROM image file may be corrupted. |
| 0xCA | Error: The SPS ME only supported recovery flash. |
| 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 the system is greater than File's DMI data length. |
| 0xD5 | Error: BIOS doesn't support EC Battery Check function. |
| 0x100 | Error: BIOS doesn't support process CPLD command information. |
| 0x101 | Error: OFBD data generate failed. |
| 0x102 | Error: Get delay time module not support. |
| 0x120 | Error: AMI BIOS Guard feature disabled. Please use AFU to flsah or enable the BIOS Guard feature in BIOS Setup. |
| 0x121 | Error: Tool does not support this BIOS Guard flash interface. |
| 0x122 | Error: ME flash not support. |
| 0x123 | Error: Runtime flash fail. |
| 0x124 | Error: Runtime flash fail. |
| 0x125 | Error: Runtime flash get status fail. |
| 0x126 | Error: Problem erasing flash. BIOS Guard is enabled, Please check the flash region is a protected region. |
| 0x127 | Error: Failed to load image into memory. BIOS Guard is enabled, This feature may not be supported. |
| 0x128 | Error: BIOS does not support ME Entire Firmware update. BIOS Guard is enabled, This feature may not be supported. |

FAQs

# The Error Message Information of ROM

AFU has added the check mechanism of ROM information since AFU 5.09.00.1284. 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.

*AFU 5.09.02.1370 or later added a warning message for informing users to decide whether continue or not when ROM information is different at the beginning.*

WARNING!!

The ROM file information does not match with the system BIOS!

If forcedly update BIOS, it may destroy the System BIOS!

We strongly do not suggest to flash the BIOS!


Press "E"- This option will update entire BIOS region and exit.

Press "A"- This option will be no ROM update.

Press "F"- This option will be forcing to follow the command by user provided.


- Please select one of the options:


AMI extremely suggest users to stop choosing "Continue to update" (Add "E" and "F" options in v5.10.00.1615 ) if users do not comprehend ROM structure very much. The system will be crashed after BIOS update because of ROM information difference.

Option "E" will flash entire blocks of BIOS region data as a new BIOS, this option also will not preserve any data.

Option "F" will keep executing the command from user input. For example, if users input command /p /b and decide to select option "F", AFU will only flash Boot block and Main block. The definition of this block will be the same as the previous Block definition.

# Windows requires a digitally signed driver
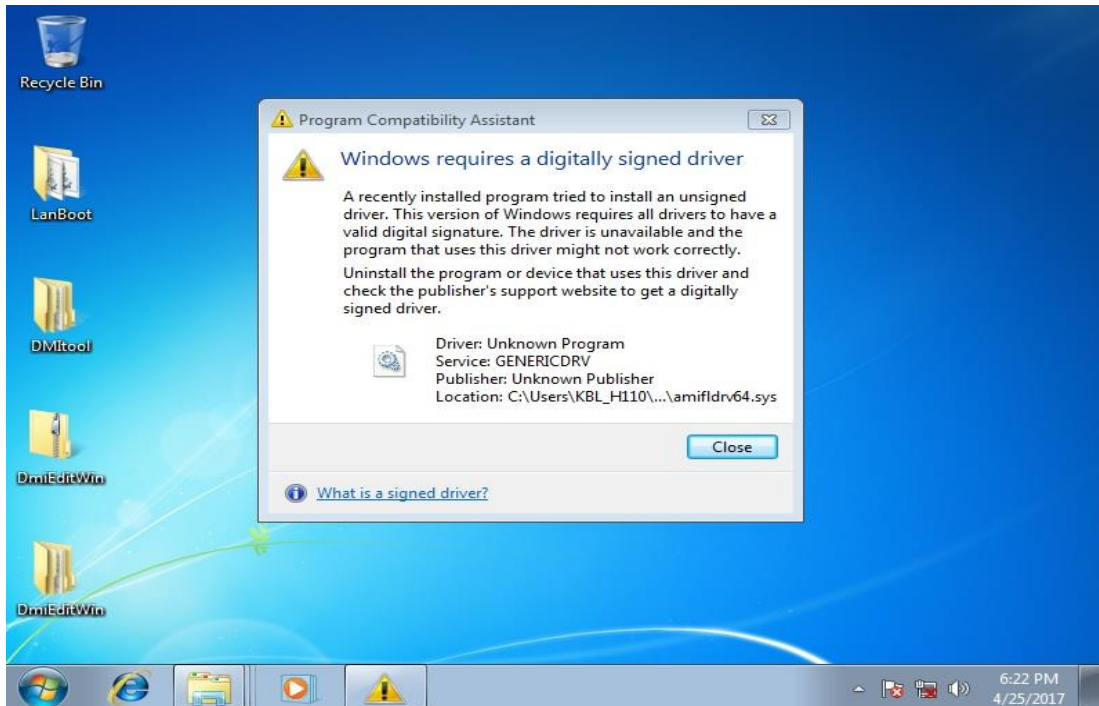


*This issue is resolved by a security fix provided by MS. KB3033929 resolves this issue. The certificate used to sign the driver is higher security and older versions of Win7 don't support it.*

# Windows version with new driver for WSMT support has to install 2 hot fixes

1. If OS is Win 7 SP1, need to install Windows hot fix, *KB3033929*.

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.

2. If OS is Win 10 build 10240 (version 1507), need to install Windows hot fix, *KB3081436*.

https://docs.microsoft.com/en-us/windows-hardware/drivers/install/driver-signing

# The former Windows driver version will not be compatible while new Windows driver version is released.

Users must use Windows driver packaged by downloaded AFU package from AMI official website or AMI SVN. AFU Windows version will detect Windows driver version whether or not is compatible. When Windows driver cannot be used via current AFU, AFU will display the error message as below image.

```
Administrator: Windows PowerShell                                    —    □    ×

F:\AfuWin64>dir
 Volume in drive F is ESD-USB
 Volume Serial Number is FE65-F469

 Directory of F:\AfuWin64

07/23/2020  11:44 AM    <DIR>          .
07/23/2020  11:44 AM    <DIR>          ..
08/05/2019  04:23 PM          631,920 AFUWINx64_5.12.02.2028.EXE
04/20/2020  11:18 AM          997,488 AFUWINx64_5.12.05.2125.EXE
11/14/2018  02:17 AM           28,544 amifldrv64.sys
               3 File(s)      1,657,952 bytes
               2 Dir(s)  14,780,768,256 bytes free

F:\AfuWin64>AFUWINx64_5.12.02.2028.EXE /s
+--------------------------------------------------------------------+
|               AMI Firmware Update Utility v5.12.02.2028            |
|       Copyright (c) 1985-2019, American Megatrends International LLC.  |
|          All rights reserved. Subject to AMI licensing agreement.  |
+--------------------------------------------------------------------+
Reading flash .............. done
 - System ROM ID = KGBIA12
 - System ROM GUID = 665b23cf-d12e-4949-83965bc626d27392
 - System ROM Secure Flash = Enable.

Process completed.

F:\AfuWin64>AFUWINx64_5.12.05.2125.EXE /s
+--------------------------------------------------------------------+
|                        AMI Utility Driver                          |
|       Copyright (c) 1985-2020, American Megatrends International LLC.  |
|          All rights reserved. Subject to AMI licensing agreement.  |
+--------------------------------------------------------------------+
Checking Driver "amifldrv" ...:
        Old driver v3.02 found - get version 4.00 or above.
10 - Error: Unable to load driver.

F:\AfuWin64>_
```

# AFU Windows version for adding additional driver description.

Because of Windows 10 secure consideration, AMI has decided to use new kernel API. However the API is not compatible with Windows 7. Therefore, to support backward compatible with Windows 7, AFU 5.14.00.0006 adds additiona driver to support above Windows 7.

If OS is above Windows 7 (eg. Windows 8.1 or 10), AFU will use the new driver.

      amigendrv32.sys

      amigendrv64.sys

If OS is Windows 7, AFU will use the original driver.

      amifldrv32.sys

      amifldrv64.sys

If OS is above Windows 7 (eg. Windows 8.1 or 10), users can rename "amifldrv64.sys" to "amigendrv64.sys" or users can directly use "amigendrv64.sys". However, If OS is Windows 7, users cannot rename "amigendrv64.sys" to "amifldrv64.sys" and use it.

# AMI AFULNX supports XEN

AFULNX v2.35 or later has supported XEN, but BIOS must add the "RuntimeMemoryHole" Module.

However, if BIOS supports WSMT feature, no need to add the module.

# Segmentation Fault when AFULNX is kernel 3.14.40 with XEN 4.2.4

Please follow steps below to operate the configuration, and try again.

1. Check if the system runs under X11, a.k.a GUI mode. If so, switch to console mode with the following command.

> # systemctl set-default multi-user.target

Remember to restart system after configured.

2. Check if Dom0 has enough free memory.

> # xl info

Check the item "free_memory", make sure it is larger than 1024. If the number is lower than 1024, use the command

# xm mem-set Domain-0 1024

You don't need to restart the system after this step.

3. Set virtual CPU number to 1 of Domain-0 in XEN.

> # xm vcpu-set Domain-0 1

4. AFULNX updates BIOS.

# "AFUxxx has stopped working" or "Segmentation Fault" error

Common reason which caused the error is that: AFU access to invalid/prohibit memory area.

Since AFU uses/needs driver file to access memory, most cases we met that caused the error are:

- (1) Don't use administrator/root privilege to execute tool.
- (2) Use unmatched version between driver file and AFU tool.
- (3) BIOS has enabled "Secure Boot" but doesn't sign driver file for it.
- (4) Use some newer/older AFU version that is not compatible with some motherboard.

For (1), if users are using:

- AFU WIN: Please open a command line with administrator privilege, then execute tool in it.

- AFU LNX: Please use "sudo" command to execute tool.

For (2), if users are using:

- AFU WIN:

    - Driver file (amifldrvXX.sys) can be found in AFU deliverables (Bundled with tool).

    - Please don't use other source of .sys file from other AFU version or other AMI tools.

- AFU LNX:

    - Driver file (amifldrv_mod.o) will be auto built by tool while executing AFU.

    - Driver file will be generated at the same folder of AFU tool.

    - Recommended to create an empty folder and put tool in it before executing, so AFU will generate the pure/clean driver file in folder.

For (3), please check "Signing Driver and Enrolling Public Key to the System" section to sign driver file.

For (4), if checked (1) (2) (3) but still see error:

- Firstly, please try the latest AFU version.

- Secondary, please try the previous AFU version (Which can run well in that motherboard previously):

    - If the previous version also fails, please check BIOS code.

    - If the previous version runs well, and also latest AFU still fails, please kindly contact AMI.