# Anleitung Making AOMEI Backupper Work from a USB Drive

I had used AOMEI Backupper Standard to capture a backup of my laptop's Windows 10 installation at an earlier time. That backup consisted of a single large .adi file. Now I wanted to restore that .adi drive image to the laptop's hard disk drive (HDD). To do that, I wanted to put AOMEI on a bootable USB flash drive. As I had found during years of using Acronis True Image Home for the same purpose, the process was quite simple: boot the laptop using AOMEI on the USB drive, and then use AOMEI to restore the .adi image, which could be located on either an internal or an external drive.

At this point, the laptop was ready; the external USB drive containing the .adi file was ready; all I lacked was the bootable USB drive containing AOMEI. This post describes the process by which I developed that USB drive. The summary is that, for purposes of booting a computer in UEFI mode, I had the best results by using AOMEI's option to produce a bootable USB (not ISO). For purposes of booting in Legacy mode, the ISOs had been good enough with other computers; but for this laptop, it seemed I might need to use AOMEI's option to add drivers to its processes of creating the ISO or the bootable USB drive.

## Background

For some years, I had used YUMI to create multiboot USB drives — that is, drives containing multiple bootable tools. YUMI provided a rudimentary menu. So I would boot the YUMI USB drive, I would view that menu, and I would select a tool to boot the computer. The selected tool might be a Linux live CD; it might be a Seagate HDD diagnostic tool; it might be Acronis or AOMEI. Really, it could be almost anything that would boot a computer for one purpose or another.

More recently, however, I had run into problems with YUMI. As discussed in a previous post, I had considered other multiboot tools. Later, I reaffirmed the conclusion that YUMI's competitors were not necessarily offering superior alternatives, though there was the possibility that the relatively complex Easy2Boot might prove to be a good solution. At this point, to some extent I had fallen back to the single-purpose solution: configure my USB drive to contain the tool needed now; reconfigure it later to contain some other tool, as the situation required. So the present post does not address multiboot considerations. Instead, Rufus seemed to be the current favorite, for purposes of creating single-boot USB drives.

The question right now was simply this: what does it take to produce a USB drive that will run AOMEI successfully? In almost all cases, that was partly a question of how to produce the necessary .iso file. The ISO format was a compressed file format that would typically contain files that would have to be unpacked and installed on the USB drive in order to make it bootable. Various programs were capable of creating ISO files that could then be loaded onto USB drives and booted. For example, a prior post describes the method by which I used ImgBurn to create an ISO from a bootable USB drive — which ISO I could then use to create other bootable USB drives. Unlike cloning or USB drive image backup (see my previous post on using AOMEI to clone a USB drive), the ISO could be combined with other ISOs on a multiboot drive. (Those prior posts contain further discussion that may be helpful to clarify aspects of the present post.)

## BIOS and UEFI

AOMEI offered the option of creating a bootable ISO. I downloaded and installed AOMEI 4.1.0 from FileHippo. Both it and the Softpedia download sites gave me a "full" version which, I feared, would prove to be shareware rather than freeware. I didn't mind paying for software. I had purchased copies of Acronis in previous years. What I did mind was finding that old backups were not restorable, for one reason or another. Since the quality of the Acronis software began to decline, and especially since starting the transition to Linux, I had been experimenting with AOMEI, Macrium, and other drive imaging alternatives. At present, I wasn't sure how this AOMEI task would turn out. We were still in the learning phase.

To create a bootable ISO, in AOMEI, I went to Utilities > Create Bootable Media > choose between Linux and Windows PE bootable disk types. AOMEI said, there, that the WinPE option was "more flexible and convenient," so I went with that.

But now we faced our first real decision: I could create either legacy or UEFI bootable disks. "Legacy" was another term for the venerable BIOS (i.e., Basic Input-Output System). UEFI was different from BIOS. But it was common (if potentially confusing) for people to refer to "the UEFI BIOS." Maybe it sounded strange to refer to "the UEFI" (pronounced in various ways, but perhaps most commonly "you-fee" or by spelling out the letters U-E-F-I). The general idea was that the computer's chips contained code ("firmware") that operated at a low level, to enable the silicon hardware to compute. This firmware would allow the hardware to pull itself up by its own bootstraps (i.e., to boot up), so as to become something more than just hardware. In everyday terminology, the question was whether my USB drive would boot the system using the UEFI BIOS or the old legacy BIOS.

How-To Geek (Hoffman, 2017) explained that UEFI had several advantages over BIOS and that Intel was phasing out the latter. One advantage was that UEFI used GPT rather than MBR partitioning, which meant that it could boot from larger (i.e., > 2.1TB) drives; those drives could have more than four primary partitions; and the data on the drive would be better protected against corruption. Among other advantages, UEFI also offered a faster boot process and a graphical user interface (GUI) with mouse support in the "BIOS" setup screen. Another post offers an image comparing BIOS and UEFI interfaces. AdamW (2014) offered more detail. To find out whether a Windows 10 computer was using BIOS or UEFI, TenForums (2017) presented several methods, including Win-R > msinfo32 > System Summary > BIOS Mode or the (elevated or boot) command *bcdedit* > look for winload.exe (BIOS) or winload.efi (UEFI).

The choice between BIOS and UEFI depended partly on the operating system — which depended, to some extent, on the hardware. A SuperUser discussion indicated that UEFI was not necessary in order to run Windows 10, but it appeared that new Windows 10 computers were almost invariably using UEFI. Microsoft offered more detailed comments on the potentially confusing question of which versions of Windows supported UEFI (e.g., "A 64-bit UEFI PC can only boot 64-bit versions of Windows. A 32-bit PC can only boot 32-bit versions of Windows. In some cases, while in legacy BIOS mode, you may be able to run 32-bit Windows on a 64-bit PC, assuming the manufacturer supports 32-bit legacy BIOS mode on the PC."). Another SuperUser comment added to that (e.g., "[A 64-bit EFI computer] can boot only 64-bit OSes in EFI mode. Thus, to boot a 32-bit version of Windows, you must boot it in BIOS mode."). Note also that, while BIOS settings were typically available by hitting a key (e.g., F2) during bootup, UEFI settings might only be available through the Windows boot menu.

## Creating the UEFI Bootable USB Drive

So I was facing a choice between creating a legacy (i.e., BIOS) or UEFI ISO in AOMEI. The information just provided makes clear that UEFI would be technically superior, but that was not really the question. The question was, rather, what will it take to boot this computer with this ISO, so that I can do what I need to do? In the present case, what I needed to do was to use AOMEI to restore an .adi image to the laptop's HDD.

It was tempting to create a legacy ISO, because YUMI would recognize it. That is, I could add the AOMEI ISO to a YUMI multiboot drive, and have all of my YUMI legacy tools together in one place. As noted in a previous post, though, I had found that an attempt to install Linux resulted in failure, when I used a legacy (BIOS) Linux (bootable) live USB on this UEFI-friendly laptop. AOMEI advised consulting the motherboard manual for guidance. It seemed clear that I should select a UEFI ISO for purposes of working with this computer.

The other question on that screen, in AOMEI, was whether I wanted to "Download WinPE creating environment from internet." The note accompanying that option said, "The program will automatically download WinPE creating environment from AOMEI server. Tips: the creating environment is base on Windows 10." I had already made the choice between the WinPE- or Linux-based versions of AOMEI. The question here was whether I wanted the WinPE version to be downloaded from the Internet, as distinct from being created from the files that were already installed with AOMEI on my Windows 10 desktop. For versions of Windows prior to Windows 10, AOMEI indicated that failing to check this box could require me to download and install the Windows AIK or ADK.

I chose the download option, and proceeded to tell AOMEI to create an ISO. AOMEI recommended several tools for purposes of getting that ISO onto a USB drive or CD. Oddly, the first two on their list were shareware; moreover, the first, UltraISO, was said to have a file size limit of 300MB, at least in its free trial. The ISO that I had just produced was 362MB. But AOMEI did also recommend two freeware programs on that list, particularly BurnAware Free. I was surprised to see that Rufus (averaging 4.4 stars from 539 raters at Softpedia) was actually surpassed by BurnAware (4.6 stars from 655 raters). So I tried it. I was a bit puzzled to see that BurnAware Free 11.4 from FileHippo was larger than BurnAware Free 11.4 from Softpedia. Softpedia had described it as freemium, while FileHippo called it simply free, so I opted to install the FileHippo version on my Windows 10 desktop. (I had to decline an offer for Avast antivirus during installation.)

Unfortunately, it seemed that BurnAware was oriented only toward creation of optical (e.g., CD, DVD) discs. So I used Rufus after all. In Rufus, I chose the newly created ISO, GPT partitioning, UEFI (non CSM) target system, FAT32 filesystem, default (4096b) cluster size, Quick format, and Create extended label and icon files. While creating the bootable USB drive, Rufus produced an odd error, considering that I had set it to format the USB drive: "Could not find primary partition on the selected disk!" But then it blew past that and finished the USB creation process. When that was done, Windows 10 Disk Management (diskmgmt.msc) had no such problem: it could see that the USB drive had only one partition, and it was a primary partition. I attempted to boot the laptop with that USB drive. With the laptop's UEFI/BIOS setup utility set to UEFI mode, it said, "No Bootable Device." In Legacy mode, likewise, it said "No bootable device — insert boot disk and press any key." These results were consistent across retries using four different USB drives. A search on that Legacy mode error message led to only a handful of English-language webpages, none of which were on target. I tried again, using other tools recommended by recent lists of best bootable USB tools, in each case choosing options like those just enumerated for Rufus, and otherwise going with the defaults:

- Universal USB Installer (UUI) 1.9.8. UUI offered AOMEI as an installation option, but would only recognize the ISO I had created if I named it “amlnx.iso.” Then it produced a weird error: “FAT32 filesystem detected. Your drive must be formatted as Fat32 or NTFS.” I clicked OK. It proceeded to install, but then it stalled with repeated messages indicating “can not open output file.” I tried again. I thought I had checked the Fat32 format box previously, but made sure to do so this time. This time, I didn’t get the same errors, but I still got “I couldn’t find a configuration file. ‘amlnx.iso’ is not supported.” So apparently I wasn’t supposed to choose that AOMEI option and use amlnx.iso. I ran UUI again, this time selecting the option to Try Unlisted Linux ISO (GRUB). A similar option had worked well in YUMI, and YUMI seemed to use the same GUI as UUI. This approach seemed to work. But when I booted the laptop, I got errors. Booting in UEFI mode, I got “No Bootable Device.” Booting in Legacy mode, I got “invalid boot indicator(0xA) for entry 0” and “partition table not recognized(chainloader_edx=0xff, err=2)” and “Error 43: … The BPB hidden_sectors should not be zero for a hard-disk partition boot sector.”
- WinSetupFromUSB (WSFU) 1.8 x64. I selected Auto-format, chose the Linux ISO/Other Grub4dos compatible ISO option, and selected the ISO. It went partway and then crashed. I tried again. I had to reformat the USB drive before WSFU would recognize it. Disk Management was unable to reformat it, but MiniTool Partition Wizard did. The second try went better. Maybe the solution, there, was to format the drive independently before running WSFU on it. Regardless, in UEFI mode, the USB drive produced the same No Bootable Device Error as above. In Legacy mode, it did produce an initial Grub4DOS menu item, but that ended with the same BPB error as with UUI (above).
- UNetbootin 6.61. I chose its Diskimage rather than its Distribution option. It would not recognize the USB drive until I clicked OK. Then it completed without problems. In the laptop, UEFI mode once again generated the No Bootable Device Error as above, while Legacy mode got a UNetbootin menu with Default as the only (and nonworking) option.

Taking a different approach, I went back to AOMEI and chose its option to create a UEFI bootable disk written directly to the USB boot device, instead of exporting an ISO file. In other words, I opted to try using AOMEI itself as the USB-creation tool. I opted not to include the several devices the installer detected on my Windows 10 desktop computer (i.e., cellphone, printer, Wi-Fi card). The resulting USB drive produced a disk read error in Legacy mode, but it ran successfully in UEFI mode. In UEFI mode, I could see that it was Windows-10 based: it used the Win10 circular hourglass substitute graphic to indicate that it was working.

Since that approach succeeded, I opted to take the step mentioned above: I wanted to produce an ISO from the USB drive, so that I would have the option of adding it to a UEFI-compatible multiboot drive at some point in the future. That task started with ImgBurn, following the steps listed in a previous post. As in that post, I used Rufus to copy that ISO onto a USB drive. In UEFI mode, that Rufus USB produced Windows Boot Manager error 0xc000000f (“The Boot Configuration Data for your PC is missing or contains errors”). In Legacy mode, it produced the familiar “No bootable device” error. Since this failure was unlike the success documented in the previous post, I suspected that it could be fixed, but I did not invest the time needed to fix it at this point.

## YUMI and the Legacy Alternative

As noted above, AOMEI offered the option of creating a legacy ISO or USB drive, and I had previously used AOMEI legacy ISOs to boot and make images of legacy (BIOS) systems. I could go back to that: in the case of this laptop, I could configure it to boot in legacy mode and, if necessary, to use MBR rather than GPT. If I did that, I could go back to using the very handy legacy-based YUMI multiboot solution.

But I wasn’t going to do that. UEFI was the way of the future — indeed, it was the way of the present, and had been increasingly so for some years now. In the case of this laptop, there was an additional reason not to go back to that. I did try booting this laptop in legacy mode using YUMI, and I succeeded: YUMI presented me with a list of several AOMEI installations, resulting from several differently configured AOMEI ISOs. The problem was that, in legacy mode, none of the relevant input devices — mouse, keyboard, or touchpad — would function. I could see the AOMEI options but, even if it was able to see my partitions larger than 2.1TB, I was not able to do anything with them. So I never got to the point of figuring out whether (as I suspected, from Linux experience) the use of legacy AOMEI to restore a system partition that would then be running in UEFI mode would cause problems.

AOMEI was familiar with this input device issue. Their suggestion was to use the “Download Win PE creating environment from internet” (above) when creating the ISO. Unfortunately, that suggestion did not fix the problem for me: the mouse, keyboard, and touchpad were still nonresponsive in AOMEI booted in legacy mode. Another approach, recorded here for posterity, was to use the suggestion offered by Sidrick in an AOMEI forum: run Double Driver (presumably on one or more computers that would have the drivers needed to make these specific input devices function properly); allow it to collect the available drivers for those input devices (along with most if not all other drivers in use on the system) into convenient folders (for e.g., Mouse, Keyboard); and then use the option to Add Drivers during the ISO (or USB) creation process, adding the contents of those folders specifically.

I did test Double Driver. It did seem to produce the desired folders containing copies of driver files. On my Windows 10 desktop, the full collection of all drivers (including e.g., modem, monitor) amounted to 618MB. Compressed, it would be less. It seemed like a good idea to run Double Driver in each Windows installation, so as to have an organized source of drivers for any purpose. For instance, in this drive imaging context, I had seen that Acronis and other programs, offering the ability to restore an image to dissimilar hardware, always wanted the user to add the drivers needed to make that hardware work. This seemed to be a convenient way of finding and organizing those drivers. Whether they would work for that purpose, or for purposes of helping AOMEI to function properly in Legacy mode, were questions that I did not explore at this point.

Quelle: https://raywoodcockslatest.wordpress.com/2018/07/25/aomei-usb/