
Switch Debian 7 wheezy from legacy to UEFI boot mode

26.10.2014

Table of Contents

1. Usage scenario	1
2. Boot a live system	2
3. Prepare the harddisk	2
3.1. Back up your data	2
3.2. Identify Debian's "/boot" partition	2
3.3. Create GPT partition table	3
3.4. Create an UEFI partition	4
4. Mount the Debian filesystems	5
5. Update debians /etc/fstab	6
6. Inside the chroot environment	6
6.1. Preparation	6
6.2. Install grub-efi	7
7. Validate the debian bootloader in UEFI Bios	8
8. References	8

1. Usage scenario

This document could be useful for you want to:

- switch my Legacy boot installation into an UEFI one,
- install a fresh Debian 7 with UEFI or
- reinstall a broken UEFI boot loader on Debian 7.

My story: I bought some time ago an Acer E3-111 netbook. The Acer E3-111 has a legacy boot mode. Although it is possible to boot and install Debian 7 wheezy in this mode, the boot process is not stable. Grub usually boots reliably but the up and down cursor keys very often do not work.

I decided to switch my Legacy boot installation into an UEFI one. ¹

Since debian 7 installers can not boot from UEFI Bios yet this howto might also help to install a fresh Debian 7 with UEFI:

1. Install debian 7 in Bios legacy mode (as usual).
2. Follow this howto.

2. Boot a live system

1. Enable UEFI in bios.
2. Boot an Ubuntu 14.4 live system on USB or DVD.

3. Prepare the harddisk

3.1. Back up your data

Back up your data!

3.2. Identify Debian's “/boot” partition

My legacy boot system had a 243MiB ext2 partition mounted on /boot. This partition is never encrypted. It is where the grub files and Linux kernels reside. Check by double clicking on the partition icon on the Ubuntu desktop and have a look inside.

```
# ls -l
total 21399
-rw-r--r-- 1 root root 155429 Sep 28 00:59 config-3.16-0.bpo.2-amd64
drwxr-xr-x 3 root root 7168 Nov 5 08:03 grub
-rw-r--r-- 1 root root 15946275 Nov 5 16:28 initrd.img-3.16-0.bpo.2-amd64
drwx----- 2 root root 12288 Nov 24 2012 lost+found
-rw-r--r-- 1 root root 2664392 Sep 28 00:59 System.map-3.16-0.bpo.2-amd64
-rw-r--r-- 1 root root 3126096 Sep 28 00:48 vmlinuz-3.16-0.bpo.2-amd64
```

```
# df -h
Filesystem      Size  Used Avail Use% Mounted on
...
```

¹After having switched to UEFI boot I had no mayor issues with Debian 7 wheezy on the Acer E3-111.
Tipp: upgrade kernel by enabling the *Debian backports* repository and configure the touchpad with `synaptics`.

```
/dev/sda1          234M   28M  206M  13% /media/....
```

As you can see in the following partition table of the Debian legacy boot system my `/boot` partition is number 1 (`/dev/sda1`).



Although 1 is the default value for standard debian installations better check!



The Ubuntu live system has identified this partition as `/dev/sda`. The debian system on your hddisk could reference it differently.

Partition table of the Debian legacy boot system

```
# fdisk -l /dev/sda
...
   Device Boot      Start         End      Blocks   Id  System
/dev/sda1  *        2048        499711       44032    7   HPFS/NTFS/exFAT
...
/dev/sda5             501760     976771071    488134656   83   Linux
```

In legacy boot mode the `/boot` partition must have the `boot` -flag (*) set. This confirms our assumption: the `/boot` filesystem is on: `/dev/sda1`.

```
# gdisk -l /dev/sda
GPT fdisk (gdisk) version 0.8.5
```

Partition table scan:

MBR: MBR only

BSD: not present

APM: not present

GPT: not present

```
...
Number  Start (sector)    End (sector)  Size      Code  Name
   1            2048          499711    243.0 MiB   8300   Linux filesystem
   5          501760     976771071    238.2 GiB   8300   Linux filesystem
```

3.3. Create GPT partition table

Transform the partition table from MBR to GPT with

```
#gdisk /dev/sda
```

`r` recovery and transformation options (experts only)
`f` load MBR and build fresh GPT from it

3.4. Create an UEFI partition

A good graphical tool is the Gnome Partition Editor `gparted`:

```
# gparted /dev/sda
```

1. Shrink the `/root` partition to 200MB in order to free 43MB (see partition 1 below).
2. Create a new 43MB partition for efi using `gparted` with partition code `EF00` (EFI system) and flag it **bootable**. Format the partition with a `fat32` ² filesystem (see partition 2 below).
3. UEFI needs additionally ³ a *not* formatted 1MB partition . ⁴ (see partition 3 below).

Leave the other partitions untouched (see partition 5 below).

Here the result:

Partition table of the Debian UEFI boot system

```
# gdisk -l /dev/sda
GPT fdisk (gdisk) version 0.8.5
```

Partition table scan:

MBR: protective
BSD: not present
APM: not present
GPT: present

Found valid GPT with protective MBR; **using GPT**.

Disk /dev/sda: 976773168 sectors, 465.8 GiB

...

Number	Start (sector)	End (sector)	Size	Code	Name
--------	----------------	--------------	------	------	------

²fat32=vfat in `/etc/fstab`

³I have not verified if the additional 1MB partition is really necessary. Omitting this step the following error message may occur: `GPT detected. Please create a BIOS-Boot partition (>1MB, unformatted filesystem, bios_grub flag)`. This can be performed via tools such as `Gparted`. Then try again.

⁴Some say it should have the flag **bios_grub**, for me it works without.

1	2048	411647	200.0 MiB	8300	Linux filesystem
2	411648	499711	43.0 MiB	EF00	Efi partition
3	499712	501759	1024.0 KiB	8300	Linux filesystem
5	501760	976771071	465.5 GiB	8300	Linux filesystem

5

4. Mount the Debian filesystems

1. Mount the `/` (root) filesystem.

- For non-encrypted root filesystems a simple `mount` will do.

```
# mount -t ext4 /dev/sda5 /mnt
```

- For encrypted root filesystems the mounting procedure can be a little tricky especially when the root filesystem resides inside a logical volume which is encrypted. Please refer to [Mount encrypted logical partition howto⁶](#) for details. The last step should look like

```
# mount -t ext4 /dev/mapper/koobue1-root /mnt
```

2. Mount the remaining filesystems.

Either this way...

```
# mount /dev/sda1 /mnt/boot
# mount /dev/sda2 /mnt/boot/efi
# for i in /dev/ /dev/pts /proc /sys ; do mount -B $i /mnt/$i ; done
```

or this way, both commands do the same...

```
# mount /dev/sda1 /mnt/boot
# mount /dev/sda2 /mnt/boot/efi
# mount --bind /sys /mnt/sys
# mount --bind /proc /mnt/proc
# mount --bind /dev /mnt/dev
# mount --bind /dev/pts /mnt/dev/pts
```

⁵ I noticed on my system the code EF00 changed somehow to 0700. Why?

⁶ [../doc/mount_encryptedLVM/mount_encrypted_logical_partition-howto.html](#)

3. Internet access

For internet access inside chroot:

```
# cp /etc/resolv.conf /mnt/etc/resolv.conf
```

5. Update debians /etc/fstab

Update the entries in `/mnt/etc/fstab` to reflect the partition changes above. Compare the UUID's there with the ones listed here:

```
# ls /dev/disk/by-uuid
```

Add the new UEFI partition (see last line in `/etc/fstab` below) in order to get it mounted permanently on `/boot/efi`.

```
# cat /mnt/etc/fstab
# <file system> <mount point> <type> <options> <dump> <pass>
/dev/mapper/koobue1-root / ext4 errors=remount-ro 0 1
# /boot was on /dev/sda1 during installation
UUID=040cdd12-8e45-48bd-822e-7b73ef9fa09f /boot ext2 defaults 0 2
/dev/mapper/koobue1-swap_1 none swap sw 0 0
/dev/sr0 /media/cdrom0 udf,iso9660 user,noauto 0 0
#Jens: tmpfs added for SSD
tmpfs /tmp tmpfs defaults,nodev,nosuid,size=500m 0
0
tmpfs /var/lock tmpfs
defaults,nodev,nosuid,noexec,mode=1777,size=100m 0 0
tmpfs /var/run tmpfs
defaults,nodev,nosuid,noexec,mode=0775,size=100m 0 0
UUID=19F0-4372 /boot/efi vfat defaults 0 2
```



I use `/dev/mapper` for the encrypted file system and `tmpfs` because I have an SSD disk.

6. Inside the chroot environment

6.1. Preparation

Enter with:

```
.....  
# chroot /mnt  
.....
```

Check

```
.....  
# cat /etc/fstab  
.....
```

for not yet mounted entries and mount them manually e.g.

```
.....  
# mount /tmp  
# mount /run  
# mount /var/lock  
...  
.....
```

6.2. Install grub-efi

```
.....  
# apt-get remove grub-pc  
# apt-get install grub-efi  
.....
```

```
.....  
# grub-install /dev/sda  
.....
```

Check presence of the efi file:

```
.....  
# file /boot/efi/EFI/debian/grubx64.efi  
/boot/efi/EFI/debian/grubx64.efi: PE32+ executable (EFI application)  
x86-64 (stripped to external PDB), for MS Windows  
.....
```

A Debian entry should be listed here:

```
.....  
# efibootmgr  
BootCurrent: 0000  
Timeout: 0 seconds  
BootOrder: 0000,2001,2002,2003  
Boot0000* debian  
Boot2001* EFI USB Device  
Boot2002* EFI DVD/CDROM  
Boot2003* EFI Network  
.....
```

Exit chroot environment.

`exit`

Reboot the system.

7. Validate the debian bootloader in UEFI Bios

The bios will not accept the bootloader by default, because `/EFI/debian/grubx64.efi` is not the default path and because the file has no Microsoft signature.

This is why `grubx64.efi` has to be validated manually in the UEFI bios setup. In my InsydeH20 bios I selected:

Security → Select an UEFI file as trusted → Enter

Then browse to

`/EFI/debian/grubx64.efi`

in order to insert the grub boot loader in the trusted bootloader bios database.



On my Acer E3-111 the bios menu entry was disabled by default. To enable it I had to define first a supervisor password.

Security → Set Supervisor Password → Enter

8. References

Tanguy

Tanguy: *Debian: switch to UEFI boot.* <http://tanguy.ortolo.eu/blog/article51/debian-efi>. April 2012.

Vulcan

Vulcan, Silviu: *Linux on the Acer E3-111 – Aspire E3-111-C5FN.* <http://www.sg-vulcan.com/linux-on-the-acer-e3-111-aspire-e3-111-c5fn/> . 09/2014.