NAME

efibootmgr - manipulate the EFI Boot Manager

SYNOPSIS

efibootmgr [**-v**]
efibootmgr -a -b *bootnum*
efibootmgr -A -b *bootnum*
efibootmgr -B -b *bootnum*
efibootmgr -c -l *loader* [**-aD**] [**-b** *bootnum*] [**-k** *kernel*] [**-L** *label*] [**-e** *env*]
efibootmgr -E [**-d**] [**-p**]
efibootmgr -F
efibootmgr -f
efibootmgr -n -b *bootnum*
efibootmgr -N
efibootmgr -o *bootorder*
efibootmgr -t *timeout*
efibootmgr -T
efibootmgr -u *unix-path*

DESCRIPTION

The **efibootmgr** program manipulates how UEFI Boot Managers boot the system. It can create and destroy methods for booting along with activating or deactivating them. It can also change the defined order of boot methods. It can create a temporary boot (BootNext) variable that references a boot method to be tried once upon the next boot.

The UEFI standard defines how hosts may control what is used to bootstrap the system. Each method is encapsulated within a persistent UEFI variable, stored by the UEFI BIOS of the form **Boot***XXXX* (where XXXX are uppercase hexadecimal digits). These variables are numbered, each describing where to load the bootstrap program from, and whether or not the method is active (used for booting, otherwise the method will be skipped). The order of these methods is controlled by another variable, **BootOrder**. The currently booted method is communicated using **BootCurrent**. A global timeout can also be set.

efibootmgr requires that the kernel module efirt(9) module be present or loaded to get and set these non-volatile variables.

The following options are available:

-a --activate
Activate the given *bootnum* boot entry, or the new entry when used with **-c**.

-A --deactivate

Deactivate the given *bootnum* boot entry.

-b --bootnum *bootnum*

When creating or modifying an entry, use *bootnum* as the index. When creating a new entry, fail if it already exists.

-B --delete

Delete the given *bootnum* boot entry.

-c --create

Create a new **Boot** variable (aka method or entry).

-D --dry-run

Process but do not change any variables.

-E --esp

Print the FreeBSD path to the ESP device, derived from the EFI variables *BootCurrent* and *BootXXXX*. This is the ESP partition used by UEFI to boot the current instance of the system. If **-d --device-path** is specified, the UEFI device path to the ESP is reported instead. If **-p --unix-path** is specified, the mount point of the ESP is reported instead.

-f --fw-ui, **-F --no-fw-ui**

Set or clear the request to the system firmware to stop in its user interface on the next boot.

-k --kernel *kernel*

The path to and name of the kernel.

-l --loader *loader*

The path to and name of the loader.

-L --label *label*

An optional description for the method.

-n --bootnext

Set *bootnum* boot entry as the **BootNext** variable.

-N --delete-bootnext

Delete the **BootNext** optional variable.

-o --bootorder *bootorder*

Set **BootOrder** variable to the given comma delimited set of *bootnum*s. The numbers are in hex to match **Boot***XXXX*, but may omit leading zeros.

-t --set-timeout *timeout*

Set the bootmenu timeout value.

-T --del-timeout

Delete the **BootTimeout** variable.

-u --efidev *unix-path*

Displays the UEFI device path of *unix-path*.

-v --verbose

Display the device path of boot entries in the output.

Examples

To display the current **Boot** related variables in the system:

```
efibootmgr -v
```

This will display the optional **BootNext** (if present), **BootCurrent** (currently booted method), followed by the optional **Timeout** value, any **BootOrder** that may be set, followed finally by all currently defined **Boot** variables, active or not. The verbose flag, (**-v**), augments this output with the disk partition uuids, size/offset and device-path of the variable. The flag will also include any unreferenced (by BootOrder) variables.

The **efibootmgr** program can be used to create new EFI boot variables. The following command may be used to create a new boot method, using the EFI partition mounted under */boot/efi*, mark the method active, using the given loader and label the method "FreeBSD-11":

```
efibootmgr -a -c -l /boot/efi/EFI/freebsd/loader.efi -L FreeBSD-11
```

This will result in the next available bootnum being assigned to a new UEFI boot variable, and given the label "FreeBSD-11" such as:

```
Boot0009 FreeBSD-11
```

Note newly created boot entries are, by default, created inactive, hence the reason **-a** flag is specified above so that it will be considered for booting. The active state is denoted by a '*' following the

Boot*XXXX* name in the output. They are also inserted into the first position of current **BootOrder** variable if it exists. They must first be set to active before being considered available to attempt booting from, else they are ignored.

```
efibootmgr -B -b 0009
```

Will delete the given boot entry Boot0009.

To set the given boot entry active:

```
efibootmgr -a -b 0009
```

To set a given boot entry to be used as the **BootNext** variable, irrespective of its active state, use:

```
efibootmgr -n -b 0009
```

To set the **BootOrder** for the next reboot use:

```
efibootmgr -o 0009,0003,...
```

SEE ALSO

efirt(9), efivar(8), gpart(8), uefi(8)

STANDARDS

The Unified Extensible Firmware Interface Specification is available from *www.uefi.org*.