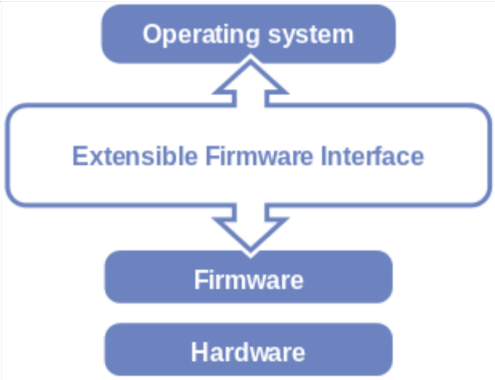
Unified Extensible Firmware Interface

From Wikipedia, the free encyclopedia

The **Unified Extensible Firmware Interface** (**UEFI**, pronounced as an initialism U-E-F-I or like "unify" without the *n*[a]) is a specification that defines a software interface between an operating system and platform firmware. UEFI replaces the Basic Input/Output System (BIOS) firmware interface originally present in all IBM PC-compatible personal computers,[2][3] with most UEFI firmware implementations providing legacy support for BIOS services. UEFI can support remote diagnostics and repair of computers, even with no operating system installed.[4]

Intel developed the original **Extensible Firmware Interface** (**EFI**) specification. Some of the EFI's practices and data formats mirror those from Microsoft Windows.[5][6] In 2005, UEFI deprecated EFI 1.10 (the final release of EFI). The Unified EFI Forum is the industry body that manages the UEFI specification.

The UEFI logo



EFI's position in the software stack

Contents

History

The original motivation for EFI came during early development of the first Intel–HP Itanium systems in the mid-1990s. BIOS limitations (such as 16-bit processor mode, 1 MB addressable space and PC AT hardware) had become too restrictive for the larger server platforms Itanium was targeting.[7] The effort to address these concerns began in 1998 and was initially called *Intel Boot Initiative*;[8] it was later renamed to EFI.[9][10]

In July 2005, Intel ceased its development of the EFI specification at version 1.10, and contributed it to the Unified EFI Forum, which has evolved the specification as the Unified Extensible Firmware Interface (UEFI). The original EFI specification remains owned by Intel, which exclusively provides licenses for EFI-based products, but the UEFI specification is owned by the Forum.[7][11]

Version 2.1 of the UEFI (*Unified Extensible Firmware Interface*) specification was released on 7 January 2007. It added cryptography, network authentication and the User Interface Architecture (Human Interface Infrastructure in UEFI). The latest UEFI specification, version 2.5, was approved in April 2015.

Advantages

The interface defined by the EFI specification includes data tables that contain platform information, and boot and runtime services that are available to the OS loader and OS. UEFI firmware provides several technical advantages over a traditional BIOS system:[12]

- Ability to boot from large disks (over 2 TB) with a GUID Partition Table (GPT)[13][b]
- CPU-independent architecture[b]
- CPU-independent drivers[b]
- Flexible pre-OS environment, including network capability
- Modular design

Compatibility

Processor compatibility

As of version 2.5, processor bindings exist for Itanium, x86, x86-64, ARM (AArch32) and ARM64 (AArch64).[14] Only little-endian processors can be supported.[15] Unofficial UEFI support is under development for POWERPC64 by implementing TianoCore on top of OPAL,[16] the OpenPOWER abstraction layer, running in little-endian mode.[17] Similar projects exist for MIPS[18] and RISC-V.[19]

Standard PC BIOS is limited to a 16-bit processor mode and 1 MB of addressable memory space, resulting from the design based on the IBM 5150 that used a 16-bit Intel 8088 processor.[7][20] In comparison, the processor mode in a UEFI environment can be either 32-bit (x86-32, AArch32) or 64-bit (x86-64, Itanium, and AArch64).[7][21] 64-bit UEFI firmware implementations support long mode, which allows applications in the preboot execution environment to use 64-bit addressing to get direct access to all of the machine's memory.[22]

UEFI requires the firmware and operating system loader (or kernel) to be size-matched; for example, a 64-bit UEFI firmware implementation can load only a 64-bit operating system boot loader or kernel. After the system transitions from "Boot Services" to "Runtime Services", the operating system kernel takes over. At this point, the kernel can change processor modes if it desires, but this bars usage of the runtime services (unless the kernel switches back again).[23]:sections 2.3.2 and 2.3.4 As of version 3.15, Linux kernel supports 64-bit kernels to be booted on 32-bit UEFI firmware implementations running on x86-64 CPUs, with *UEFI handover* support from a UEFI boot loader as the requirement.[24] UEFI handover protocol deduplicates the UEFI initialization code between the kernel and UEFI boot loaders, leaving the initialization to be performed only by the Linux kernel's *UEFI boot stub*.[25][26]

Disk device compatibility

In addition to the standard PC disk partition scheme that uses a master boot record (MBR), UEFI also works with a new partitioning scheme called GUID Partition Table (GPT), which is free from many of the limitations of MBR. In particular, the MBR limits on the number and size of disk partitions (up to four primary partitions per disk, and up to 2 TiB (2×2^{40} bytes) per disk) are relaxed.[27] More specifically, GPT allows for a maximum disk and partition size of 8 ZiB (8×2^{70} bytes).[27][28]

Linux

Support for GPT in Linux is enabled by turning on the option `CONFIG_EFI_PARTITION` (EFI GUID Partition Support) during kernel configuration.[29] This option allows Linux to recognize and use GPT disks after the system firmware passes control over the system to Linux.

For reverse compatibility, Linux can use GPT disks in BIOS-based systems for both data storage and booting, as both GRUB 2 and Linux are GPT-aware. Such a setup is usually referred to as *BIOS-GPT*.[30] As GPT incorporates the protective MBR, a BIOS-based computer can boot from a GPT disk using GPT-aware boot loader stored in the protective MBR's bootstrap code area.[28] In case of GRUB, such a configuration requires a BIOS Boot partition for GRUB to embed its second-stage code due to absence of the post-MBR gap in GPT partitioned disks (which is taken over by the GPT's *Primary Header* and *Primary Partition Table*). Commonly 1 MiB in size, this partition's Globally Unique Identifier (GUID) in GPT scheme is `21686148-6449-6E6F-744E-656564454649` and is used by GRUB only in BIOS-GPT setups. From the GRUB's perspective, no such partition type exists in case of MBR partitioning. This partition is not required if the system is UEFI-based because no embedding of the second-stage code is needed in that case.[13][28][30]

UEFI systems can access GPT disks and boot directly from them, which allows Linux to use UEFI boot methods. Booting Linux from GPT disks on UEFI systems involves creation of an EFI System partition (ESP), which contains UEFI applications such as bootloaders,

operating system kernels, and utility software.[31][32][33] Such a setup is usually referred to as *UEFI-GPT*, while ESP is recommended to be at least 512 MiB in size and formatted with a FAT32 filesystem for maximum compatibility.[28][30][34]

For backward compatibility, most UEFI implementations also support booting from MBR-partitioned disks, through the Compatibility Support Module (CSM) that provides legacy BIOS compatibility.[35] In that case, booting Linux on UEFI systems is the same as on legacy BIOS-based systems.

Microsoft Windows

The 64-bit versions of Microsoft Windows Vista[c] and later, 32-bit versions of Windows 8 and later, and the Itanium versions of Windows XP and Server 2003 can boot from disks with a partition size larger than 2 TB.

Features

Services

EFI defines two types of services: *boot services* and *runtime services*. Boot services are available only while the firmware owns the platform (i.e., before the `ExitBootServices` call), and they include text and graphical consoles on various devices, and bus, block and file services. Runtime services are still accessible while the operating system is running; they include services such as date, time and NVRAM access.

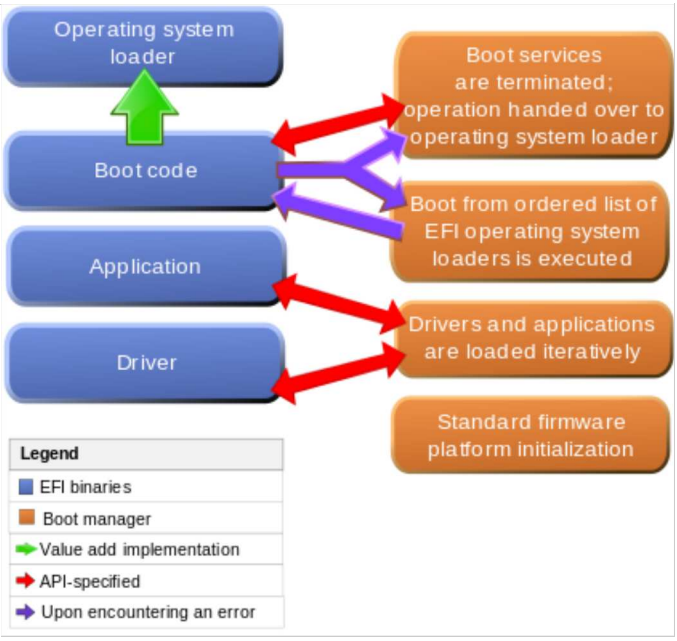
In addition, the *Graphics Output Protocol* (GOP) provides limited runtime services support; see also Graphics features section below. The operating system is permitted to directly write to the framebuffer provided by GOP during runtime mode. However, the ability to change video modes is lost after transitioning to runtime services mode until the OS graphics driver is loaded.

Variable services
: UEFI variables provide a way to store data, in particular non-volatile data, that is shared between platform firmware and operating systems or UEFI applications. Variable namespaces are identified by GUIDs, and variables are key/value pairs. For example, variables can be used to keep crash messages in NVRAM after a crash for the operating system to retrieve after a reboot.[36]

Time services
: UEFI provides device-independent time services. Time services include support for timezone and daylight saving fields, which allow the hardware real-time clock to be set to local time or UTC.[37] On machines using a PC-AT real-time clock, the clock still has to be set to local time for compatibility with BIOS-based Windows.[6]

Applications

Independently of loading an operating system, UEFI has the ability to run standalone *UEFI applications*, which can be developed and installed independently of the system manufacturer. UEFI applications reside as files on the ESP and can be started directly by the firmware's boot manager, or by other UEFI applications. One class of the UEFI applications are the operating system loaders, such as rEFInd, Gummiboot, and Windows Boot Manager; they start a specific operating system and optionally provide a user interface for the selection of another UEFI application to run. Utilities like the UEFI shell are also UEFI applications.



Interaction between the EFI boot manager and EFI drivers

Protocols

EFI defines protocols as a set of software interfaces used for communication between two binary modules. All EFI drivers must provide services to others via protocols.

Device drivers

In addition to standard architecture-specific device drivers, the EFI specification provides for a processor-independent device driver environment, called *EFI byte code* or *EBC*. System firmware is required by the UEFI specification to carry an interpreter for any EBC images that reside in or are loaded into the environment. In that sense, EBC is analogous to Open Firmware, the hardware-independent firmware used in PowerPC-based Apple Macintosh and Sun Microsystems SPARC computers, among others.

Some architecture-specific (non-EBC) EFI device driver types can have interfaces for use from the operating system. This allows the operating system to rely on EFI for basic graphics and network functions until operating-system-specific drivers are loaded.

Graphics features

The EFI specification defined a UGA (Universal Graphic Adapter) protocol as a way to support device-independent graphics. UEFI did not include UGA and replaced it with GOP (Graphics Output Protocol), with the explicit goal of removing VGA hardware dependencies. The two are similar.[38]

UEFI 2.1 defined a "Human Interface Infrastructure" (HII) to manage user input, localized strings, fonts, and forms (in the HTML sense). These enable original equipment manufacturers (OEMs) or independent BIOS vendors (IBVs) to design graphical interfaces for pre-boot configuration; UEFI itself does not define a user interface.

Most early UEFI firmware implementations were console-based, but as early as 2007 some implementations featured a graphical user interface.[39]

EFI System partition

EFI System partition, often abbreviated as ESP, is a data storage device partition that is used in computers adhering to the UEFI specification. Accessed by the UEFI firmware when a computer is powered up, it stores UEFI applications and the files these applications need to run, including operating system kernels. Supported partition table schemes include MBR and GPT, as well as El Torito volumes on optical disks.[23]:section 2.6.2 For the use on ESPs, UEFI defines a specific version of the FAT file system, which is maintained as part of the UEFI specification and independently from the original FAT specification, encompassing a variant of the FAT32 file system on ESPs, and FAT16 and FAT12 file systems on removable media.[23]:section 12.3[40][41] The ESP also provides space for a boot sector as part of the backward BIOS compatibility.[35]

Booting

UEFI booting

Unlike BIOS, UEFI does not rely on a boot sector, defining instead a boot manager as part of the UEFI specification. When a computer is powered on, the boot manager checks the boot configuration and, based on its settings, loads and executes the specified operating system loader or operating system kernel. The boot configuration is a set of global-scope variables stored in NVRAM, including the boot variables that indicate the paths to operating system loaders or kernels, which as a component class of UEFI applications are stored as files on the firmware-accessible EFI system partition (ESP).

Operating system loaders can be automatically detected by a UEFI implementation, which enables easy booting from removable devices such as USB flash drives. This automated detection relies on a standardized file path to the operating system loader, with the path depending on the computer architecture. The format of the file path is defined as `<EFI_SYSTEM_PARTITION>/BOOT/BOOT<MACHINE_TYPE_SHORT_NAME>.EFI`; for example, the file path to the loader on an x86-64 system is `/efi/BOOT/BOOTX64.EFI`.[23]

Booting UEFI systems from GPT-partitioned disks is commonly called *UEFI-GPT booting*. It is also common for a UEFI implementation to include a menu-based user interface to the boot manager, allowing the user to manually select the desired operating system (or system utility) from a list of available boot options.

CSM booting

To ensure backward compatibility, most UEFI firmware implementations on PC-class machines also support booting in legacy BIOS mode from MBR-partitioned disks, through the *Compatibility Support Module (CSM)* that provides legacy BIOS compatibility. In this scenario, booting is performed in the same way as on legacy BIOS-based systems, by ignoring the partition table and relying on the content of a boot sector.[35]

BIOS-style booting from MBR-partitioned disks is commonly called *BIOS-MBR*, regardless of it being performed on UEFI or legacy BIOS-based systems. Furthermore, booting legacy BIOS-based systems from GPT disks is also possible, and such a boot scheme is commonly called *BIOS-GPT*.

Despite the fact that the UEFI specification requires MBR partition tables to be fully supported,[23] some UEFI firmware implementations immediately switch to the BIOS-based CSM booting depending on the type of boot disk's partition table, effectively preventing UEFI booting to be performed from EFI System partitions on MBR-partitioned disks.[35] Such a boot scheme is commonly called *UEFI-MBR*.

Network booting

The UEFI specification includes support for booting over network via the Preboot eXecution Environment (PXE). Underlying network protocols include Internet Protocol (IPv4 and IPv6), User Datagram Protocol (UDP), Dynamic Host Configuration Protocol (DHCP) and Trivial File Transfer Protocol (TFTP).[23][42]

Also included is support for accessing boot images remotely stored on storage area networks (SANs), with Internet Small Computer System Interface (iSCSI) and Fibre Channel over Ethernet (FCoE) as supported protocols for accessing the SANs.[23][43][44]

Version 2.5 of the UEFI specification adds support for accessing boot images over the HTTP protocol.[45]

Secure boot

The UEFI 2.3.1 Errata C specification (or higher) defines a protocol known as Secure Boot, which can secure the boot process by preventing the loading of drivers or OS loaders that are not signed with an acceptable digital signature.[46] When secure boot is enabled, it is initially placed in "setup" mode, which allows a public key known as the "Platform key" (PK) to be written to the firmware. Once the key is written, secure boot enters "User" mode, where only drivers and loaders signed with the platform key can be loaded by the firmware. Additional "Key Exchange Keys" (KEK) can be added to a database stored in memory to allow other certificates to be used, but they must still have a connection to the private portion of the Platform key.[47] Secure boot can also be placed in "Custom" mode, where additional public keys can be added to the system that do not match the private key.[48]

Secure boot is supported by Windows 8 and 8.1, Windows Server 2012 and 2012 R2, and a number of Linux distributions including Fedora (since version 18), openSUSE (since version 12.3), and Ubuntu (since version 12.04.2).[49] As of June 2015, FreeBSD support is in a planning stage.[50]

Compatibility Support Module

The *Compatibility Support Module* (CSM) is a component of the UEFI firmware that provides legacy BIOS compatibility by emulating a BIOS environment, allowing legacy operating systems and some option ROMs that do not support UEFI to still be used.[51]

CSM also provides required legacy System Management Mode (SMM) functionality, called *CompatibilitySmm*, as an addition to features provided by the UEFI SMM. This is optional, and highly chipset and platform specific. An example of such a legacy SMM functionality is providing USB legacy support for keyboard and mouse, by emulating their classic PS/2 counterparts.[51]

UEFI shell

UEFI provides a shell environment, which can be used to execute other UEFI applications, including UEFI boot loaders.[33] Apart from that, commands available in the UEFI shell can be used for obtaining various other information about the system or the firmware, including getting the memory map (`memmap`), modifying boot manager variables (`bcfg`), running partitioning programs (`diskpart`), loading UEFI drivers, and editing text files (`edit`).[52][53][54]

Source code for a UEFI shell can be downloaded from the Intel's TianoCore UDK2010 / EDK2 SourceForge project.[55] Shell v2 works best in UEFI 2.3+ systems and is recommended over the shell v1 in those systems. Shell v1 should work in all UEFI systems.[52][56][57]

Methods used for launching UEFI shell depend on the manufacturer and model of the system motherboard. Some of them already provide a direct option in firmware setup for launching, e.g. compiled x86-64 version of the shell needs to be made available as `<EFI_SYSTEM_PARTITION>/SHELLX64.EFI`. Some other systems have an already embedded UEFI shell which can be launched by appropriate key press combinations.[58][59] For other systems, the solution is either creating an appropriate USB flash drive or adding manually (`bcfg`) a boot option associated with the compiled version of shell.[54][58][60][61]

Extensions

Extensions to EFI can be loaded from virtually any non-volatile storage device attached to the computer. For example, an original equipment manufacturer (OEM) can distribute systems with an EFI partition on the hard drive, which would add additional functions to the standard EFI firmware stored on the motherboard's ROM.

Implementation and adoption

Intel EFI

Intel's implementation of EFI is the *Intel Platform Innovation Framework*, codenamed *Tiano*. Tiano runs on Intel's XScale, Itanium and IA-32 processors, and is proprietary software, although a portion of the code has been released under the BSD license or Eclipse Public License (EPL) as **TianoCore**. TianoCore can be used as a payload for coreboot.[62]

Phoenix Technologies' implementations of UEFI include its SecureCore and SecureCore Tiano products.[63] American Megatrends offers its own UEFI firmware implementation known as Aptio,[64] while Insyde Software offers InsydeH2O, its own implementation of Tiano.[65]

Platforms using EFI/UEFI

Intel's first Itanium workstations and servers, released in 2000, implemented EFI 1.02.

Hewlett-Packard's first Itanium 2 systems, released in 2002, implemented EFI 1.10; they were able to boot Windows, Linux, FreeBSD and HP-UX; OpenVMS added UEFI capability in June, 2003.

In January 2006, Apple Inc. shipped its first Intel-based Macintosh computers. These systems used EFI instead of Open Firmware, which had been used on its previous PowerPC-based systems.[66] On 5 April 2006, Apple first released Boot Camp, which produces a Windows drivers disk and a non-destructive partitioning tool to allow the installation of Windows XP or Vista without requiring a reinstallation of Mac OS X. A firmware update was also released that added BIOS compatibility to its EFI implementation. Subsequent Macintosh models shipped with the newer firmware.[67]

During 2005, more than one million Intel systems shipped with Intel's implementation of UEFI.[68] New mobile, desktop and server products, using Intel's implementation of UEFI, started shipping in 2006. For instance, boards that use the Intel 945 chipset series use Intel's UEFI firmware implementation.

Since 2005, EFI has also been implemented on non-PC architectures, such as embedded systems based on XScale cores.[68]

The EDK (EFI Developer Kit) includes an NT32 target, which allows EFI firmware and EFI applications to run within a Windows application. But no direct hardware access is allowed by EDK NT32. This means only a subset of EFI application and drivers can be executed at the EDK NT32 target.

In 2008, more x86-64 systems adopted UEFI. While many of these systems still allow booting only the BIOS-based OSes via the Compatibility Support Module (CSM) (thus not appearing to the user to be UEFI-based), other systems started to allow booting UEFI-based OSes. For example, IBM x3450 server, MSI motherboards with ClickBIOS, all HP EliteBook Notebook and Tablet PCs, newer HP Compaq Notebook PCs (e.g., 6730b, 6735b, etc.).

In 2009, IBM shipped System x machines (x3550 M2, x3650 M2, iDataPlex dx360 M2) and BladeCenter HS22 with UEFI capability. Dell shipped PowerEdge T610, R610, R710, M610 and M710 servers with UEFI capability. More commercially available systems are mentioned in a UEFI whitepaper.[69]

In 2011, major vendors (such as ASRock, Asus, Gigabyte, and MSI) launched several consumer-oriented motherboards using the Intel 6-series LGA 1155 chipset and AMD 9 Series AM3+ chipsets with UEFI.[70]

With the release of Windows 8 in October 2012, Microsoft's certification requirements now require that computers include firmware that implements the UEFI specification. Furthermore, if the computer supports the "Connected Standby" feature of Windows 8 (which allows devices to have power management comparable to smartphones, with an almost instantaneous return from standby mode), then the firmware is not permitted to contain a Compatibility Support Module (CSM). As such, systems that support Connected Standby are incapable of booting Legacy BIOS operating systems.[71][72]

Operating systems

An operating system that can be booted from a (U)EFI is called a (U)EFI-aware operating system, defined by (U)EFI specification. Here the term *booted from a (U)EFI* means directly booting the system using a (U)EFI operating system loader stored on any storage device. The default location for the operating system loader is `<EFI_SYSTEM_PARTITION>/BOOT/BOOT<MACHINE_TYPE_SHORT_NAME>.EFI`, where short name of the machine type can be `IA32`, `X64`, `IA64`, `ARM` or `AA64`.[23] Some operating systems vendors may have their own boot loaders. They may also change the default boot location.

- The Linux kernel has been able to use EFI at boot time since early 2000,[73] using the elilo EFI boot loader or, more recently, EFI versions of GRUB.[74] Grub+Linux also supports booting from a GUID partition table without UEFI.[13] The distribution Ubuntu added support for UEFI secure boot as of version 12.10.[75] Further, the Linux kernel can be compiled with the option to run as an EFI bootloader on its own through the EFI bootstub feature.
- HP-UX has used (U)EFI as its boot mechanism on IA-64 systems since 2002.
- HP OpenVMS has used (U)EFI on IA-64 since its initial evaluation release in December 2003, and for production releases since January 2005.[76]
- Apple uses EFI for its line of Intel-based Macs. Mac OS X v10.4 Tiger and Mac OS X v10.5 Leopard implement EFI v1.10 in 32-bit mode even on newer 64-bit CPUs, but full support arrived with Mac OS X v10.8 Mountain Lion.[77]
- The Itanium versions of Windows 2000 (Advanced Server Limited Edition and Datacenter Server Limited Edition) implemented EFI 1.10 in 2002. MS Windows Server 2003 for IA-64, MS Windows XP 64-bit Edition and Windows 2000 Advanced Server Limited Edition, all of which are for the Intel Itanium family of processors, implement EFI, a requirement of the platform through the DIG64 specification.[78]
- Microsoft introduced UEFI for x86-64 Windows operating systems with Windows Server 2008 and Windows Vista Service Pack 1 so the 64-bit versions of Windows 7 are compatible with EFI. However, Compatibility Support Module (CSM) may still need to be turned on in UEFI mode to install Windows 7 and Windows Vista in UEFI mode to GPT partitions because of some option ROMs being different between native UEFI mode (without CSM) and hybrid UEFI mode (with CSM). 32-bit UEFI was originally not supported since vendors did not have any interest in producing native 32-bit UEFI firmware because of the mainstream status of 64-bit computing.[79] Windows 8 includes further optimizations for UEFI systems, including a faster startup, 32-bit UEFI support, and secure boot support.[80][81]
- On 5 March 2013, the FreeBSD Foundation awarded a grant to a developer seeking to add UEFI support to the FreeBSD kernel and bootloader.[82] The changes were initially stored in a discrete branch of the FreeBSD source code, but were merged into the mainline source on 4 April 2014 (revision 264095); the changes include support in the installer as well.[83]

- Oracle Solaris 11.1 and later support UEFI boot for x86 systems with UEFI firmware version 2.1 or later. GRUB 2 is used as the boot loader on x86.[84]

Use of UEFI with virtualization

- HP Integrity Virtual Machines provides UEFI boot on HP Integrity Servers. It also provides a virtualized UEFI environment for the guest UEFI-aware OSes.
- Intel hosts an Open Virtual Machine Firmware project on SourceForge.[85]
- VMware Fusion 3 software for Mac OS X can boot Mac OS X Server virtual machines using UEFI.
- VMware Workstation prior to version 11 unofficially supports UEFI, but is manually enabled by editing the .vmx file.[86] VMware Workstation version 11 and above supports UEFI, independently of whether the physical host system is UEFI-based. As of January 2016, no version of VMware Workstation supports the Secure Boot feature of UEFI.[87]
- The vSphere ESXi 5.0 hypervisor officially support UEFI.[88]
- VirtualBox has implemented UEFI since 3.1,[89] but limited to Unix/Linux operating systems (does not work with Windows Vista x64 and Windows 7 x64).[90][91]
- QEMU can be used with the Open Virtual Machine Firmware (OVMF) provided by TianoCore.[92]
- The VMware ESXi version 5 hypervisor, part of VMware vSphere, supports virtualized UEFI as an alternative to BIOS inside a virtual machine.
- The second generation of the Microsoft Hyper-V virtual machine supports virtualized UEFI.[93]

Applications development

EDK2 Application Development Kit (EADK) makes it possible to use standard C library functions in UEFI applications. EADK can be freely downloaded from the Intel's TianoCore UDK2010 / EDK2 SourceForge project. As an example, a port of the Python interpreter is made available as a UEFI application by using the EADK.[94]

A minimalistic "hello, world" C program written using EADK looks similar to its usual C counterpart:

```
#include <Uefi.h>
#include <Library/UefiLib.h>
#include <Library/ShellCEntryLib.h>

EFI_STATUS EFIAPI ShellAppMain(IN UINTN Argc, IN CHAR16 **Argv)
{
    Print(L"hello, world\n");
    return EFI_SUCCESS;
}
```

Criticism

Numerous digital rights activists have protested against UEFI. Ronald G. Minnich, a co-author of coreboot, and Cory Doctorow, a digital rights activist, have criticized EFI as an attempt to remove the ability of the user to truly control the computer.[95][96] It does not solve any of the BIOS's long-standing problems of requiring two different drivers—one for the firmware and one for the operating system—for most hardware.[97]

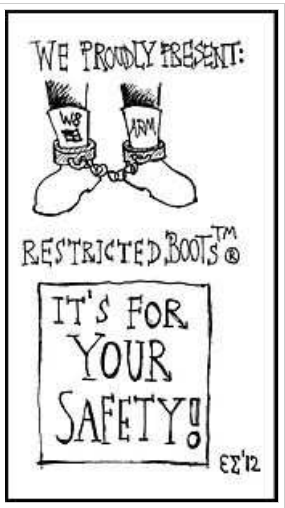
Open-source project TianoCore also provides the UEFI interfaces.[98] TianoCore lacks the specialized drivers that initialize chipset functions, which are instead provided by Coreboot, of which TianoCore is one of many payload options. The development of Coreboot requires cooperation from chipset manufacturers to provide the specifications needed to develop initialization drivers.

Secure boot

In 2011, Microsoft announced that computers certified to run its Windows 8 operating system had to ship with secure boot enabled using a Microsoft private key. Following the announcement, the company was accused by critics and free software/open source advocates (including the Free Software Foundation) of trying to use the secure boot functionality of UEFI to hinder or outright prevent the installation of alternative operating systems such as Linux. Microsoft denied that the secure boot requirement was intended to serve as a form of lock-in, and clarified its requirements by stating that Intel-based systems certified for Windows 8 must allow secure boot to enter custom mode or be disabled, but not on systems using the ARM architecture.[48][100]

Other developers raised concerns about the legal and practical issues of implementing support for secure boot on Linux systems in general. Former Red Hat developer Matthew Garrett noted that conditions in the GNU General Public License version 3 may prevent the use of the GRUB bootloader without a distribution's developer disclosing the private key (however, the Free Software Foundation has since clarified its position, assuring that the responsibility to make keys available was held by the hardware manufacturer),[75] and that it would also be difficult for advanced users to build custom kernels that could function with secure boot enabled without self-signing them.[100] Other developers suggested that signed builds of Linux with another key could be provided, but noted that it would be difficult to persuade OEMs to ship their computers with the required key alongside the Microsoft key.[3]

Several major Linux distributions have developed different implementations for secure boot. Matthew Garrett himself developed a minimal bootloader known as a shim, which is a precompiled, signed bootloader that allows the user to individually trust keys provided by distributors.[101] Ubuntu 12.10 uses an older version of shim pre-configured for use with Canonical's own key that verifies only the bootloader and allows unsigned kernels to be loaded; developers believed that the practice of signing only the bootloader is more feasible, since a trusted kernel is effective at securing only the user space, and not the pre-boot state for which secure boot is designed to add protection. That also allows users to build their own kernels and use custom kernel modules as well, without the need to reconfigure the system.[75][102][103] Canonical also maintains its own private key to sign installations of Ubuntu pre-loaded on certified OEM computers that run the operating system, and also plans to enforce a secure boot requirement as well—requiring both a Canonical key and a Microsoft key (for compatibility reasons) to be included in their firmware. Fedora also uses shim, but requires that both the kernel and its modules be signed as well.[102]



Winner of the Free Software Foundation's Restricted Boot webcomic contest[99]

It has been disputed whether the kernel and its modules must be signed as well; while the UEFI specifications do not require it, Microsoft has asserted that their contractual requirements do, and that it reserves the right to revoke any certificates used to sign code that can be used to compromise the security of the system.[103] In February 2013, another Red Hat developer attempted to submit a patch to the Linux kernel that would allow it to parse Microsoft's authenticode signing using a master X.509 key embedded in PE files signed by Microsoft. However, the proposal was criticized by Linux creator Linus Torvalds, who attacked Red Hat for supporting Microsoft's control over the secure boot infrastructure.[104]

On 26 March 2013, the Spanish free software development group Hispalinux filed a formal complaint with the European Commission, contending that Microsoft's secure boot requirements on OEM systems were "obstructive" and anti-competitive.[105]

At the Black Hat conference in August 2013, a group of security researchers presented a series of exploits in specific vendor implementations of UEFI that could be used to exploit secure boot.[106]

Windows 10 will allow OEMs to not offer the ability to configure or disable secure boot on x86 systems.[107]

Firmware issues

The increased prominence of UEFI firmware in devices has also led to a number of technical issues blamed on their respective implementations.[108]

Following the release of Windows 8 in late 2012, it was discovered that certain Lenovo computer models with secure boot had firmware that was hardcoded to allow only executables named "Windows Boot Manager" or "Red Hat Enterprise Linux" to load, regardless of any other setting.[109] Other issues were encountered by several Toshiba laptop models with secure boot that were missing certain certificates required for its proper operation.[108]

In January 2013, a bug surrounding the UEFI implementation on some Samsung laptops was publicized, which caused them to be bricked after installing a Linux distribution in UEFI mode. While potential conflicts with a kernel module designed to access system features on Samsung laptops were initially blamed (also prompting kernel maintainers to disable the module on UEFI systems as a safety measure), Matthew Garrett uncovered that the bug was actually triggered by storing too many UEFI variables to memory, and that the bug could also be triggered under Windows as well under special conditions. In conclusion, he determined that the offending kernel module had caused kernel message dumps to be written to the firmware, thus triggering the bug.[36][110][111]

See also

- Advanced Configuration and Power Interface (ACPI)
- Open Firmware
- OpenBIOS
- Platform Initialization Specification
- System Management BIOS (SMBIOS)
- System Management Mode (SMM)
- Trusted Platform Module (TPM)
- Unified EFI Forum

Notes

a. Various pronunciations have existed for UEFI; according to the UK *PC Pro Magazine*, the following pronunciations are in use: "weffy" (PC Pro), "U-E-F-I" (Microsoft), "you-fee", and "you-ef-fee". The magazine also notes the lack of agreement on the pronunciation.[1]
b. Large disk support and features such as Advanced Configuration and Power Interface (ACPI) and System Management BIOS (SMBIOS) were subsequently implemented in BIOS-based systems.
c. For Microsoft Windows Vista (x64), it is possible only if installed from an installation DVD of Microsoft Windows Vista (x64) with its service pack 1 or 2 integrated.

References

1. "UEFI BIOS Explained". *PCPro.co.uk*. 2013-05-03. Retrieved 2014-07-05.
2. Kinney, Michael (1 September 2000). "Solving BIOS Boot Issues with EFI" (PDF). pp. 47–50. Retrieved 14 September 2010.
3. "MS denies secure boot will exclude Linux". The Register. 23 September 2011. Retrieved 24 September 2011.
4. "The 30-year-long Reign of BIOS is Over: Why UEFI W... - Input Output". *HP.com*. Archived from the original on 2013-06-26. Retrieved 2012-03-06.
5. IBM PC Real Time Clock should run in UT (https://www.cl.cam.ac.uk/~mgk25/mswish/ut-rtc.html). Cl.cam.ac.uk. Retrieved on 2013-10-30.
6. Garrett, Matthew (19 January 2012). "EFI and Linux: The Future Is Here, and It's Awful". *linux.conf.au 2012*. Retrieved 2 April 2012.
7. "Emulex UEFI Implementation Delivers Industry-leading Features for IBM Systems" (PDF). Emulex. Retrieved 14 September 2010.
8. *Extensible Firmware Interface (EFI) and Unified EFI (UEFI)*, Intel, archived from the original on 2010-01-05
9. Wei, Dong (2006), "foreword", *Beyond BIOS*, Intel Press, ISBN 978-0-9743649-0-2
10. "1.10 Specification overview", *Extensible Firmware Interface*, Intel
11. *About*, Unified EFI Forum, "Q: What is the relationship between EFI and UEFI? A: The UEFI specification is based on the EFI 1.10 specification published by Intel with corrections and changes managed by the Unified EFI Forum. Intel still holds the copyright on the EFI 1.10 specification, but has contributed it to the Forum so that the Forum can evolve it. There will be no future versions of the EFI specification, but customers who license it can still use it under the terms of their license from Intel. The license to the Unified EFI Specification comes from the Forum, not from Intel"
12. "UEFI and Windows". Microsoft. 15 September 2009. Retrieved 14 September 2010.
13. "Installation". *3.4 BIOS installation*. GNU GRUB. Retrieved 2013-09-25.
14. UEFI Specification 2.4, section 2.3
15. UEFI specification 2.3.1, section 1.8.1.
16. "GitHub - andreiw/ppc64le-edk2: TianoCore UEFI for OPAL/PowerNV (PPC64/PowerPC64 Little-Endian)". *GitHub*.
17. "Tianocore for OpenPOWER". *Firmware Security*.
18. kontais. "EFI-MIPS". *SourceForge*.
19. "lowRISC · lowRISC".
20. Hardwidge, Ben (1 June 2010). "LBA explained — Solving the 3TB Problem?". bit-tech. Retrieved 18 June 2010.
21. Brian Richardson (10 May 2010). "Ask a BIOS Guy: "Why UEFI" ". Intel Architecture Blog. Retrieved 18 June 2010.
22. Gary Simpson. "UEFI Momentum — The AMD perspective". AMD. Archived from the original (PPTX) on 2014-01-04. Retrieved 2014-09-20.
23. "UEFI Specifications (version 2.4 and older)" (PDF). Unified EFI, Inc. June 2013. Retrieved 2013-09-25.
24. "Linux kernel 3.15, Section 1.3. EFI 64-bit kernels can be booted from 32-bit firmware". *kernelnewbies.org*. 2014-06-08. Retrieved 2014-06-15.
25. "x86, efi: Handover Protocol". LWN.net. 2012-07-19. Retrieved 2014-06-15.
26. "Linux kernel documentation: Documentation/efi-stub.txt". kernel.org. 2014-02-01. Retrieved 2014-06-15.
27. "FAQ: Drive Partition Limits" (PDF). UEFI Forum. Retrieved 9 June 2010.
28. Roderick W. Smith (2012-07-03). "Make the most of large drives with GPT and Linux". IBM. Retrieved 2013-09-25.
29. "block/partitions/Kconfig (3.11.1)". *CONFIG_EFI_PARTITION (line #247)*. kernel.org. Retrieved 2013-09-25.
30. "GRUB". *BIOS systems*. Arch Linux. Retrieved 2013-09-25.
31. "GRUB and the boot process on UEFI-based x86 systems". *redhat.com*. Retrieved 2013-11-14.
32. "UEFI Booting 64-bit Redhat Enterprise Linux 6". *fpmurphy.com*. September 2010. Retrieved 2013-11-14.
33. "UEFI Bootloaders". *archlinux.org*. Retrieved 2013-09-25.
34. "Unified Extensible Firmware Interface: EFI System Partition". *archlinux.org*. Retrieved 2013-09-25.
35. "UEFI system booting from MBR partition table and GRUB legacy". Arch Linux Forums. June 2012. Retrieved 2013-10-06.
36. "Samsung UEFI bug: Notebook bricked from Windows". The H. Retrieved 27 February 2013.
37. UEFI specification, section 7.3
38. "Intel Embedded Graphics Drivers FAQ: BIOS and firmware". Intel. Retrieved 2014-05-19.
39. Intel shows PC booting Windows with UEFI firmware (http://apcmag.com /5862/intel_shows_pc_booting_windows_with_uefi_firmware)
40. "UEFI Specification Version 2.5, Section 12.3 File System Format" (PDF). *uefi.org*. April 2015. pp. 536, 537. Retrieved 2015-05-29. "The file system supported by the Extensible Firmware Interface is based on the FAT file system. EFI defines a specific version of FAT that is explicitly documented and testable. Conformance to the EFI specification and its associate reference documents is the only definition of FAT that needs to be implemented to support EFI. To differentiate the EFI file system from pure FAT, a new partition file system type has been defined."
41. "Technical Note TN2166: Secrets of the GPT". *developer.apple.com*. 2006-11-06. Retrieved 2015-05-06.
42. "Red Hat Enterprise Linux 6 Installation Guide". *30.2.2. Configuring PXE boot for EFI*. Red Hat. Retrieved 2013-10-09.
43. "UEFI Summit" (PDF). *Advances in Pre-OS Networking in UEFI 2.4*. Hewlett-Packard. July 2013. Retrieved 2013-10-09.
44. "Storage and Network Convergence Using FCoE and iSCSI" (PDF). IBM. July 2012. Retrieved 2013-10-09.
45. "New UEFI HTTP Boot support in UEFI 2.5". *firmwaresecurity.com*. 2015-05-09. Retrieved 2015-08-13.
46. "Secure Boot Overview". Microsoft. Retrieved 18 February 2016.
47. Edge, Jake. "UEFI and "secure boot" ". LWN.net. Retrieved 9 September 2012.
48. "Windows 8 Secure Boot: The Controversy Continues". PC World. Retrieved 9 September 2012.
49. Matthew Garrett (2012-12-27). "Secure Boot distribution support". Mjg59.dreamwidth.org. Retrieved 2014-03-20.
50. "SecureBoot - FreeBSD Wiki". FreeBSD. Retrieved 16 June 2015.
51. "Intel® Platform Innovation Framework for EFI" (PDF). *Compatibility Support Module Specification (revision 0.97)*. Intel. 2007-09-04. Retrieved 2013-10-06.
52. "Unified Extensible Firmware Interface". *UEFI Shell*. Arch Linux. Retrieved 2013-09-25.
53. "EFI Shells and Scripting". Intel. Retrieved 2013-09-25.
54. "UEFI Shell Specification Version 2.0, Errata A" (PDF). Unified EFI, Inc. May 2012. Retrieved 2013-09-25.
55. "TianoCore on SourceForge". Intel. Retrieved 2013-09-25.
56. "Email Archive: edk2-devel". *[edk2] Inclusion of UEFI shell in Linux distro iso*. SourceForge. 2012. Retrieved 2013-09-25.
57. "TianoCore on SourceForge". *Shell FAQ*. Intel. Retrieved 2013-09-25.
58. "Unified Extensible Firmware Interface". *Launching UEFI Shell*. Arch Linux. Retrieved 2013-09-25.
59. "Basic Instructions for Using EFI for Server Configuration on Intel® Server Boards and Intel® Server Systems" (PDF). Intel. 2008. Retrieved 2013-09-25.
60. "Unified Extensible Firmware Interface". *bcfg*. Arch Linux. Retrieved 2013-09-25.
61. "GRUB EFI Examples". *Asus*. Arch Linux. Retrieved 2013-09-25.
62. "TianoCore - coreboot". Retrieved 25 May 2012.
63. "SecureCore Tiano™". Phoenix Technologies. Retrieved 14 September 2010.
64. "Aptio®: The Complete UEFI Product Solution" (PDF). American Megatrends, Inc. Retrieved 8 January 2011.
65. "InsydeH2O UEFI Framework". Insyde Software Corp. Retrieved 8 January 2011.
66. Apple Computer. "Universal Binary Programming Guidelines, Second Edition: Extensible Firmware Interface (EFI) (http://developer.apple.com/documentation/MacOSX/Conceptual /universal_binary/universal_binary_diffs /chapter_3_section_10.html)"

67. Apple's Transition from Open Firmware to Extensible Firmware Interface (http://www.mactech.com/articles/mactech/Vol.23/23.05/OpenFirmwareToEFI/index.html), mactech, 2007.
68. "Intel® Platform Innovation Framework for UEFI Overview". Intel. Retrieved 14 September 2010.
69. *Evaluating UEFI using Commercially Available Platforms and Solutions* (PDF), UEFI, May 2011
70. Asus P67 Motherboard Preview (http://www.bit-tech.net/hardware/motherboards/2010/11/16/asus-lga1155-motherboard-preview/1).
71. "Windows Hardware Certification Requirements for Client and Server Systems". Microsoft. January 2013. "System.Fundamentals.Firmware.CS.UEFISecureBoot.ConnectedStandby ... Platforms shall be UEFI Class Three (see UEFI Industry Group, Evaluating UEFI using Commercially Available Platforms and Solutions, version 0.3, for a definition) with no Compatibility Support Module installed or installable. BIOS emulation and legacy PC/AT boot must be disabled."
72. "Microsoft: All You Need to Know About Windows 8 on ARM". *PC Magazine*. Retrieved 30 September 2013.
73. Announcement of release 3.5pre1 (http://sourceforge.net/mailarchive/forum.php?thread_name=1077918495.14506.1.camel%40raphael.fc.hp.com&forum_name=elilo-announce) by maintainer Brett Johnson made on 2004-02-27.
74. *EFI version of Grub*, Debian GNU/Linux, retrieved 1 May 2008
75. "Ubuntu will use GRUB 2 for its Secure Boot implementation". The H Online. Retrieved 28 October 2012.
76. *OpenVMS Release History*, HP, retrieved 16 September 2008
77. *rEFIt — Windows Vista and EFI*, SourceForge
78. "Extensible Firmware Interface", *Windows Server TechCenter*, Microsoft
79. "Unified Extended Firmware Interface support in Windows Vista". Microsoft. 26 October 2006. Retrieved 12 June 2010. "Microsoft determined that vendors would not have any interest in producing native UEFI 32-bit firmware because of the current status of mainstream 64-bit computing and platform costs. Therefore, Microsoft originally did not to ship support for 32-bit UEFI implementations."
80. "Microsoft Touts Incredible Windows 8 Boot Times". Retrieved 9 September 2011.
81. Jon Brodkin (21 September 2011). "Windows 8 secure boot could complicate Linux installs". Ars Technica. Retrieved 23 September 2011.
82. "FreeBSD to get UEFI support". The H. Retrieved 7 March 2013.
83. "UEFI - FreeBSD Wiki". FreeBSD.org. Retrieved 19 June 2014.
84. "Oracle Solaris 11.1 — What's New" (PDF). oracle.com. Retrieved 2013-11-04.
85. *Open Virtual Machine Firmware*, SourceForge
86. "VMWare Workstation EFI firmware | VMware Communities". Communities.vmware.com. Retrieved 2014-02-28.
87. "Using EFI/UEFI firmware in a VMware Virtual Machine | VMware Communities". Communities.vmware.com. Retrieved 2016-01-18.
88. "What's New in vSphere 5.0". Vmware.com. Retrieved 2014-02-28.
89. *3.1 Changelog*, VirtualBox
90. *Ticket 7702*, VirtualBox
91. "Statement by sr. software engineer at Oracle", *Forum*, VirtualBox
92. "Testing secureboot with KVM". FedoraProject. Retrieved 2014-02-28.
93. "What's New in Hyper-V for Windows Server 2012 R2". MicrosoftTechNet. Retrieved 2013-06-24.
94. "TianoCore on SourceForge: EDK2 Application Development Kit (EADK)". Intel. Retrieved 2013-09-25.
95. "Interview: Ronald G Minnich". Fosdem. 6 February 2007. Retrieved 14 September 2010.
96. Doctorow, Cory (2011-12-27), *The Coming War on General Purpose Computation*, retrieved 2013-09-25
97. "coreboot (aka LinuxBIOS): The Free/Open-Source x86 Firmware". YouTube. 31 October 2008. Retrieved 14 September 2010.
98. "Welcome", *TianoCore*, SourceForge
99. Joshua Gay (2012-07-05). "FSF announces winner of Restricted Boot webcomic contest". *fsf.org*. Retrieved 2015-08-14.
100. "Is Microsoft Blocking Linux Booting on ARM Hardware?". Compurer World UK. Retrieved 2012-03-06.
101. "Shimming your way to Linux on Windows 8 PCs". ZDNet. Retrieved 26 February 2013.
102. "Ubuntu details its UEFI secure boot plans". Linux Weekly News. Retrieved 11 September 2012.
103. "No Microsoft certificate support in Linux kernel says Torvalds". The H. Retrieved 26 February 2013.
104. "Linus Torvalds: I will not change Linux to "deep-throat Microsoft" ". Ars Technica. Retrieved 26 February 2013.
105. "Exclusive: Open software group files complaint against Microsoft to EU". Reuters. 26 March 2013. Retrieved 26 March 2013.
106. "Researchers demo exploits that bypass Windows 8 Secure Boot". *IT World*. Retrieved 5 August 2013.
107. "Windows 10 to make the Secure Boot alt-OS lock out a reality". *Ars Technica*. Retrieved 21 March 2015.
108. "Linux on Windows 8 PCs: Some progress, but still a nuisance". ZDNet. Retrieved 26 February 2013.
109. "Lenovo UEFI Only Wants To Boot Windows, RHEL". Phoronix. Retrieved 26 February 2013.
110. "Linux acquitted in Samsung laptop UEFI deaths". Bit-tech. Retrieved 26 February 2013.
111. "Booting Linux using UEFI can brick Samsung laptops". The H. Retrieved 26 February 2013.

Further reading

- Zimmer, Vincent; Rothman, Michael; Hale, Robert (10 May 2007). "EFI Architecture". *Dr. Dobb's Journal*. UBM. Retrieved 12 October 2012.
- De Boyne Pollard, Jonathan (11 July 2011). "The EFI boot process". *Frequently Given Answers*. Retrieved 12 October 2012.
- De Boyne Pollard, Jonathan (8 December 2011). "The Windows NT 6 boot process". *Frequently Given Answers*. Retrieved 12 October 2012.
- Smith, Roderick W. (2011). "A BIOS to UEFI Transformation". *Roderick W. Smith's Web Page*. Retrieved 12 October 2012.
- Kothari, Rajiv (21 September 2011). "UEFI – Just How Important It Really Is". *Hardware Secrets*. Retrieved 12 October 2012.
- Fisher, Doug (2011). "UEFI Today: Bootstrapping the Continuum". *Intel Technology Journal* (Intel) **15** (01). ISBN 9781934053430. Retrieved 2013-09-24.

External links

- Official website (http://www.uefi.org/)
- Intel-sponsored open-source EFI Framework initiative (http://sourceforge.net/apps/mediawiki/tianocore/). SourceForge.
- Intel EFI/UEFI portal (http://www.intel.com/content/www/us/en/architecture-and-technology/unified-extensible-firmware-interface/efi-homepage-general-technology.html)
- UEFI documents (http://feishare.com/uefi/page-3)
- Microsoft UEFI Support and Requirements for Windows Operating Systems (http://msdn.microsoft.com/en-us/windows/hardware/gg463144.aspx)

Wikimedia Commons has media related to ***Extensible Firmware Interface***.

- How Windows 8 Hybrid Shutdown / Fast Boot feature works (http://www.techrepublic.com/blog/windows-and-office/how-windows-8-hybrid-shutdown-fast-boot-feature-works/)
- Securing the Windows 8 Boot Process (http://technet.microsoft.com/en-us/windows/dn168167.aspx)

Categories: BIOS | Firmware | Unified Extensible Firmware Interface

---