Introducing UEFI-Compliant Firmware on IBM System x and BladeCenter Servers

Revision 1.2

Nathan C. Skalsky,
Cecil Lockett,
Michael Turner,
Adam L. Soderlund,
Michael Brinkman,
with System x UEFI/BIOS Development Team
IBM Systems and Technology Group

Executive overview

This paper introduces and describes key features of the Unified Extensible Firmware Interface (UEFI) firmware in IBM® System x® and BladeCenter® servers. The following x86 IBM servers include native UEFI support (as well as BIOS compatibility), as of the date of this document:

- IBM iDataPlex™ dx360 M2
- IBM BladeCenter HS22/HS22v
- IBM System x3200/3250 M3
- IBM System x3400/3500 M2-M3
- IBM System x3550/3650 M2-M*3*
- IBM System x3850 X5

UEFI firmware replaces the basic input/output system (BIOS) of previous-generation servers while maintaining full backward compatibility.

Overview of IBM UEFI firmware

The UEFI firmware in IBM servers is a UEFI 2.1 compliant next-generation firmware that enables more cross-brand commonality, improved power and systems management, reliability and predictive fault technologies, and operating-system deployment options. The UEFI firmware replaces the BIOS of previous-generation System x and x86 BladeCenter offerings.

The UEFI firmware (like BIOS) is primarily responsible for initializing the essential system hardware (such as memory, microprocessors, and PCI buses), publishing operating-system-required data structures and functions, initializing boot devices, and handing off (booting) to an operating-system boot loader.

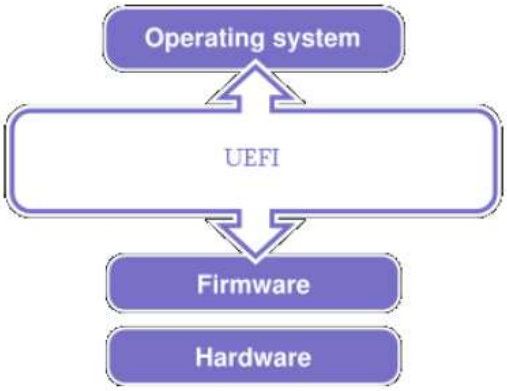
The IBM UEFI firmware has many features that go well beyond the basic requirements of UEFI-compliant firmware, including the following key features:

- Ability to boot UEFI-compliant operating systems and existing legacy master boot record operating systems without requiring that boot modes or settings be changed
- Active Energy Manager
- Enhanced light path diagnostics
- Fatal memory error recovery and DIMM isolation
- Memory predictive failure technology
- Out-of-band configuration and deployment capabilities
- Simplified hierarchical POST and UEFI diagnostic codes

Compatibility is a cornerstone of IBM server design. Therefore, the UEFI firmware is designed to support UEFI capabilities and features and BIOS. UEFI-based System x servers are capable of booting BIOS-booted operating systems as well as UEFI operating systems and can use and boot from legacy adapters as well as UEFI-compliant adapters.

Introduction to UEFI

The Unified Extensible Firmware Interface (UEFI) replaces the basic input/output system (BIOS), which has enabled x86 personal computer and server products for nearly 30 years, starting with the IBM Personal Computer (PC) in 1981. UEFI defines a standard interface between the operating system, platform firmware, and external devices and offers capabilities that far exceed those of BIOS and improved effectiveness of server development.



Intel® Corporation developed EFI in the mid 1990s to enable the Itanium® class of processors without having to implement legacy processor operating modes that are required by BIOS. Later, the UEFI Forum Inc. was formed by Intel, IBM, Hewlett-Packard, Dell, Apple, and other companies to turn EFI into a industry standard that is appropriate for broader use on other architectures, including x86 and x64.

In 2001, IBM introduced xSeries 450 and 455 (Itanium) servers with an EFI-compliant firmware. Today, UEFI 2.1 is becoming widely accepted throughout the industry, and many systems are making the transition to UEFI-compliant firmware.

The UEFI firmware enables the following code to be invoked from the UEFI preboot environment:

- **Operating-system boot loaders:** A special type of UEFI application that invokes and passes control to a protected-mode operating system.
- **UEFI applications:** Files that contain architecture-specific binary machine code or EFI byte code that can be loaded into memory and invoked. Common examples include the UEFI shell (command-line environment) and third-party adapter tools and updates.

Limitations of BIOS

BIOS originated with the first IBM PC in 1981 and has grown over time to accommodate the ever-increasing demands of enterprise and high-performance computing. Significant architectural limitations restrict the further growth of BIOS. One major architectural restraint is that BIOS runs in 16-bit processor mode, causing the following functionality limitations:

- Generally, only 1 MB of memory is addressable at any time.
- The space for PCI option ROMs and how much code they can run are limited, restricting both the number of adapters that can be installed (approximately four) and how much functionality they can contain.
- Space for advanced BIOS functionality is limited.
- The firmware image is monolithic and non-modular.

Although much of the x86 processor, memory, and I/O technology has greatly evolved over the past 30 years, the system BIOS has remained essentially unchanged.

The original BIOS was not intended to accommodate server technologies such as scalable multi-way systems, advanced power and energy management and capping, remote console, and systems management.

Overview of the IBM System x POST/boot process

The boot process consists of four phases:

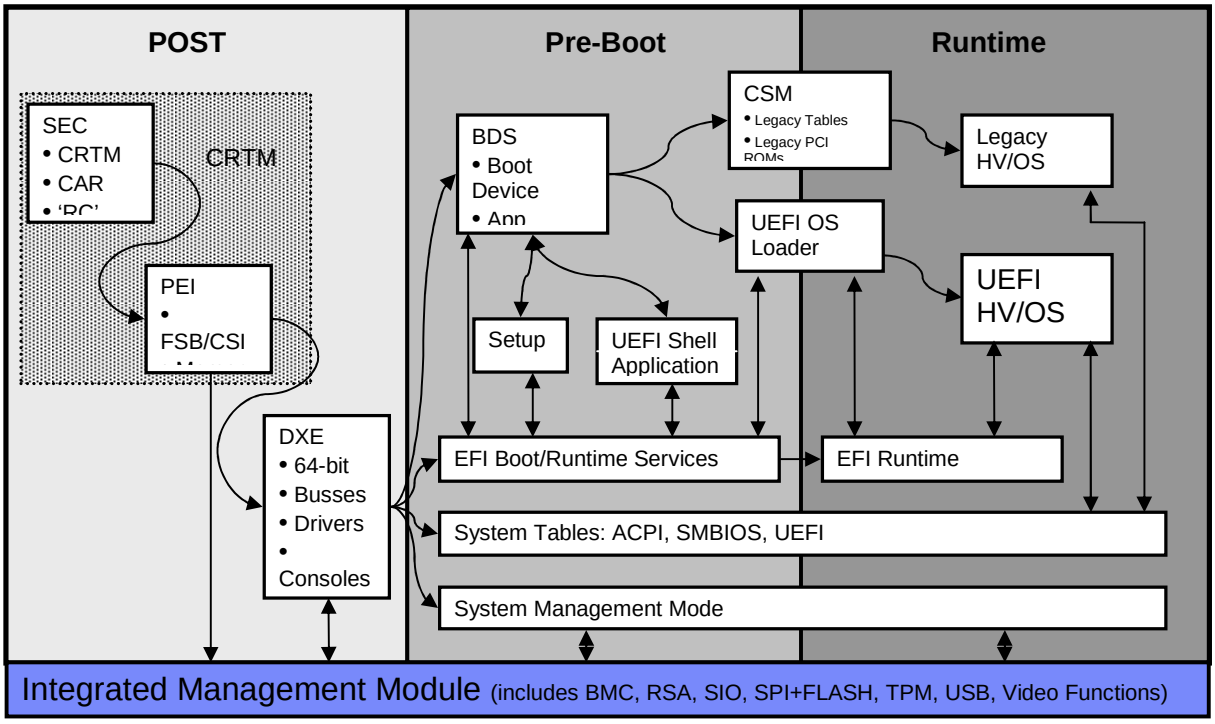
1. **Integrated management module initialization (IMM) and platform power sequence:** The IMM initializes and instructs the power sequencer to begin powering the server hardware, concluding with the processors reaching the reset vector (where they start executing UEFI firmware code).
2. **Platform initialization:** RAM is initialized and tested, processor links are trained, and basic chip set initialization is completed.
3. **Preboot:** Driver execution environment (DXE) and UEFI device drivers are discovered, loaded, and dispatched. These modular device drivers provide basic and advanced functionality. Most device drivers are loaded from the firmware flash, and some come from UEFI-compliant adapters.
4. **Boot device selection:** The user can run the Setup utility or manually select a boot option. Otherwise, the firmware iterates the list of boot targets until it finds one that exists. **Note:** A BIOS boot cannot return to UEFI (without rebooting).

Integrated management module boot process dependency

Starting with Intel Nehalem-EP-based servers, the baseboard management controller and optional Remote Supervisor Adapter II have been replaced by an integrated management module (IMM), which provides a greater set of systems-management functions, power optimization functions, and remote management options. This embedded controller is an integral part of the firmware preboot process; therefore, the IMM must fully initialize before the server can complete preboot and start the operating-system boot process.

Early power control

In most modular and tower System x servers, beginning with 4Q 2009 UEFI firmware and IMM firmware updates, the capability exists to power on the server while the IMM is still initializing. This does not change the total time that is required to start the operating system, but it does enable you to activate the server before the IMM is fully initialized. With early power control, the server initializes the primary video console and displays IMM-initialization progress messages. When the IMM initialization is complete, the server finishes the preboot and operating-system boot process normally.



Two boot methods, one firmware

IBM UEFI firmware supports UEFI booting and BIOS booting without requiring that boot modes or settings be changed. Unlike in other UEFI/BIOS compatible systems, IBM UEFI firmware supports both boot methods within one managed boot order.

The UEFI boot process is significantly different from the BIOS boot process: instead of booting devices, the system boots specific targets from physical or logical device paths, so boot media can contain virtually unlimited boot targets (operating-system instances).

By default, the firmware favors UEFI boot targets if both are available; for example, if the CD drive is first in the boot list and a dual bootable media (booted BIOS or UEFI) is present, the firmware detects that the CD is UEFI-bootable and hands off to the EFI boot loader. You can override this preference by inserting the **Legacy Only** flag above a boot target in the boot list that you want to boot BIOS only.

Notes:

- If you insert the **Legacy Only** flag into the boot order list, the server invokes the compatibility module for booting regardless of whether there appears to be anything there to boot (that is, UEFI is not aware of media behind non-UEFI compliant adapters). When the Legacy Only option is invoked, the server is committed to a BIOS boot, and the server eventually resets if nothing is booted during the BIOS boot process.
- Some UEFI-aware operating systems that also have BIOS support are installed according to how the dual-boot installation media was booted. Therefore, if you want to install Microsoft® Windows® Server 2008 x64 as a UEFI-aware operating system, you must boot the UEFI boot loader on the installation media. This happens by default unless you use the **Legacy Only** flag to force the media to be booted as a BIOS boot target.

The boot process

The UEFI specification extends the boot process to allow several new boot management capabilities. Key features of UEFI include the ability to create user-defined boot options, associate additional parameters to pass to the boot loader, assign unique names (for example, “Emma’s Maintenance OS”), and manage multiple operating-system installations on one boot device.

The following table compares the boot processes and highlights some of the differences between BIOS booting and UEFI booting.

BIOS boot process	UEFI boot process
Hands off control to the master boot record (MBR)	Hands off control to a boot loader on the UEFI partition
One MBR per boot device	Possible multiple boot loaders on one device
Options are limited to categories such as CD or hard disk drive.	User-defined boot options can be created in addition to generic boot device categories.
Configuration information cannot be passed to the MBR and operating system.	The user or operating-system agent can add parameters to the boot options.
The boot order information is stored in CMOS memory.	Boot options and their order are stored in NVRAM.

Notes:

- IBM UEFI firmware supports both boot processes through the System x boot manager with the limitation that BIOS booting is terminal; that is, when the compatibility support module (CSM) is invoked for a boot, the system cannot return to the UEFI boot manager.
- In UEFI bootable media, the main UEFI boot loader is usually at \efi\boot\bootx64.efi. It must be at that location if the firmware is to boot it automatically without requiring that you manually add a boot option for where the applicable boot loader binary is located and

named. Any valid UEFI boot loader at efi\boot\bootx64.efi is invoked when the boot option for the device is enumerated.

Generic boot options

The generic boot options are, for the most part, smart options. If a CD or DVD contains a UEFI boot record in the master catalog, the CD/DVD boot is an EFI boot. If the CD or DVD does not contain a UEFI boot record but it does contain an x86 boot record, the CD/DVD boot is a BIOS boot.

Dual-boot media is media that can simultaneously meet UEFI and BIOS boot requirements.

Generic boot option	Dual-boot media support
CD or DVD	Yes*
Diskette	Yes*
USB storage	Yes*
Hard disk 0,1, 2, 3, 4	No; either master boot record (MBR) or GUID partition table (GPT)
Network	N/A; will try EFI network boot first*
Embedded hypervisor	N/A; hard-wired to boot the hypervisor option

* If there is no **Legacy Only** flag in the boot order list, the boot manager gives preference to booting dual-boot media as UEFI. If you insert the **Legacy Only** flag above a boot target in the boot order list, that media will be forced into a BIOS boot.

Note: All UEFI bootable media must be FAT formatted.

Legacy Only is not a boot option but a flag that indicates to the firmware that any generic boot options below it in the boot order list are to be BIOS booted, even if they could be UEFI booted. There are two designed reasons for the Legacy Only option:

- To force a CD/DVD BIOS installation for a dual-boot CD or DVD
- To boot to a hard disk that is not visible to the EFI environment (a non-IBM disk controller without EFI firmware)

CD or DVD

A table of contents describes partitions and indicates whether those partitions are bootable and, if they are, which architecture (BIOS x86 or UEFI).

Notes:

- The Microsoft Windows Server 2008 installation DVD contains two bootable partitions: one for BIOS x86 and one for UEFI.
- Some versions of SLES 11 installation media contain two bootable partitions; however, both are designated for BIOS x86 and therefore will be booted through the BIOS compatibility mechanism.

Diskette

Diskettes use a FAT file system, and they are not able to store a large amount of data. However, a diskette is tested to determine whether it is bootable, and if it is, the server goes into BIOS mode to boot the diskette. Note that a formatted diskette always appears to be bootable, even if it has no DOS or other bootable image; therefore, the presence of a diskette in the drive causes the server to attempt to boot from it. Remove any diskette from the drive before you boot the server unless you intend to boot from it.

USB storage

USB storage is similar to a hard disk in that it has a master boot record (MBR). However, because it is removable media, the specification allows for USB storage to include a \efi\boot\bootx64.efi file. If a USB key contains the file, it is EFI bootable. You can place a fullshell.efi file on a USB key and rename it to \efi\boot\bootx64.efi. Then, if you boot from the USB key, it EFI boots to the shell. If there is no \efi\boot\bootxX64.efi file on the USB key, the UEFI boot manager examines the MBR, and if it is designated as bootable, the server goes into BIOS mode and boots from the USB key. (See "Starting the UEFI shell" for information about obtaining and running the shell environment.)

Hard disk

A BIOS-bootable operating system can be installed only to a master boot record (MBR) partition, and an EFI-bootable operating system can be installed only to a GUID partition table (GPT) partition. MBR partitions and GPT partitions may not coexist on the same volume. Therefore, if a selected partition is MBR, it is BIOS-booted; if it is GPT, it is EFI-booted.

Network

The server first attempts an EFI network boot. If that fails, the server passes control to the compatibility source module (CSM) to attempt a BIOS network boot. You can configure the iSCSI and PXE settings through the Setup utility (**System Settings → Network panel**). You can specify **Legacy**, **UEFI**, **Both**, or **None**.

Note: A UEFI firmware update that is available in 4Q of 2009 might be required to enable network boots of UEFI operating systems to boot without specific boot options. This might happen in deployment scenarios in which the booting system is not the one on which the operating system was installed. This generic media path boot support is specified only for removable media in the UEFI 2.1 specification; it has been generalized in the UEFI 2.3 specification.

Embedded hypervisor

A hypervisor is a USB key with a specific vendor ID/product ID (VID/PID). It is excluded from processing in the USB storage entry, and USB keys are excluded when the Embedded Hypervisor entry is processed. An embedded hypervisor key can be installed internally or externally.

Operating-system deployment

The UEFI boot manager processes devices in the boot order list, one at a time, as it searches for a potential bootable device. If the boot manager detects a UEFI-bootable device, it attempts to boot that device. If that attempt fails, the boot manager returns to the boot order list. For most devices, the boot manager goes to the next device and tries again. For PXE and iSCSI, the boot manager checks for Legacy at this time.

If the firmware detects that the boot target is only BIOS bootable (that is, it has only a bootable MBR), it will take the current boot order list and perform the same step as Legacy Only, and transition to legacy mode.

Example:
The boot order is as follows:

1. DVD
2. Hard disk drive 0 (formatted with a bootable BIOS MBR)
3. USB

The UEFI boot manager inspects for a DVD and determines that no DVD is present.

Next, the boot manager attempts to inspect hard disk drive 0. If the boot manager successfully inspects hard disk drive 0 and determines that it is has a bootable BIOS MBR, the boot manager sends the boot order list to the compatibility support module (CSM), unloads UEFI from memory, and starts the CSM for a BIOS boot. However, if the boot manager is unable to inspect hard disk drive 0 (for example, because it is managed by an atypical legacy storage adapter that does not have a UEFI device driver and cannot have UEFI support emulated through thunking), the boot manager does not recognize a BIOS operating system. In this case, you must manually add the **Legacy Only** flag above **HardDrive0** to instruct the boot manager to assume that a BIOS operating system is associated with hard disk drive 0.

For information about adapter support issues and use of the **Legacy Only** flag, see “BIOS support” and “Optimizing boot-time performance.”

Notes:

- Before you attempt to install a UEFI-aware operating system on a hard disk that is already formatted with an MBR, you must delete all partitions from the disk or reformat the disk with GPT.
- The UEFI 2.4 specification allows the UEFI boot manager to look in \efi\boot\bootx64.efi for a valid UEFI boot loader on a hard disk drive when no specific boot loader is specified in the boot option, because a generic boot option (such as HardDrive0) or a partial media path is being used. Earlier UEFI specifications provide this capability only for removable media such as USB keys.

Boot-management limitations

This section describes the boot-management limitations that are associated with the UEFI firmware and how to work around those limitations.

Non-UEFI compliant boot devices

Some non-UEFI-compliant adapters, such as storage controllers, provide UEFI emulation interface wrappers (known as legacy adapter thunking) through the IBM UEFI firmware. A controller adapter that is not UEFI compliant (that is, it does not have a UEFI device driver) and is not enabled for legacy adapter thunking requires special consideration when you configure the server to boot from targets that are managed by that controller. You must insert the **Legacy Only** flag above the generic boot option that represents the target adapter. For more information, see “Using the Legacy Only flag.” Inserting the **Legacy Only** flag instructs the UEFI firmware to invoke the compatibility support module (CSM) and attempt a legacy boot, even though it cannot detect a boot target. The CSM detects the boot target because the legacy option ROM will have run in this case.

Generic boot options

Generic boot options provide BIOS-like ease of configuration; however, they do not allow you to specify which specific device in a category to boot first. To overcome this limitation, you can reorder the option ROM execution order (for legacy boots from adapters) or add specific device path boot options (for UEFI boots).

Multi-path controllers

Although you can use the Setup utility to effectively create UEFI and BIOS boot options, there are cases in which it is preferable to use the boot option that is created by a UEFI-aware operating system (for example, multi-path controllers). Manually adding a boot option by traversing the file system and adding the boot loader results in a specific device path, whereas the operating system can create a media path that allows a multi-path controller to lose a path and still boot.

Multiple USB storage devices

When you use the generic USB boot option, the server boots the first enumerated bootable USB device that it detects. If two or more bootable USB keys are connected, you cannot control which one the server boots.

Configuring servers for automated UEFI deployments

The ability to remotely boot deployment tools is important to efficient deployments in mid-sized and large enterprises. The following methods are available for deployment to IBM System x servers:

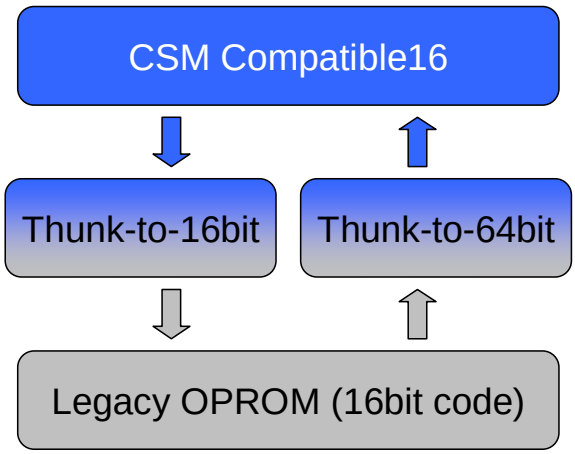
- Remotely booting a UEFI application or shell (a specific and common UEFI application)
- Using remote USB media through the integrated management module (IMM) or using local USB or DVD media:
  1. Using the Advanced Settings Utility (locally or remotely) or the Setup utility (locally or remotely through remote presence), add **USB** to the boot order list, above the first bootable target.
  2. Prepare the media or ISO image:
     a. Format the media with a FAT file system.
     b. Rename the UEFI binary file to bootx64.efi and place it in the efi\boot path. This instructs the firmware to boot the application target, even without a specific boot option that would contain an arbitrary path and file name of the binary file.
     c. (Optional) Place auto-scripting files as needed. For the UEFI shell, if a file named startup.nsh is found in the efi\boot path, it is automatically executed after the shell is loaded.
  3. (Remote media only) Using the IMM remote control application, remotely mount a USB image that contains the UEFI application that you want to boot. This might require that you purchase an IBM Virtual Media Key.
  4. (Local media only) Insert a USB key into an available USB connector on the target server.
  5. Power on the servers remotely or locally.

BIOS support

The IBM Surepath® BIOS compatibility support module (CSM) is packaged with IBM UEFI firmware, allowing for time-proven BIOS support. The CSM allows non-UEFI-compliant adapters and boot devices to be used as part of the boot process and to boot non-UEFI-compliant operating systems. It also provides the following functionality:

- It allows the use of specific PCI adapters that have only a legacy option ROM to function in the UEFI preboot environment. (this compatibility support is limited to PCI storage or video devices).
- It allows the installation and use of non-UEFI-compliant (BIOS) operating systems.

The CSM provides the ability for IBM UEFI firmware implementations to use and interact with devices that are controlled by legacy option ROMs and to boot to a BIOS operating system. UEFI firmware does this by creating a 16-bit BIOS environment that contains the necessary data structures and interfaces to allow legacy code to run. The CSM is not a full BIOS implementation; it provides the



Thunking explained in the context of executing legacy code from a legacy option ROM

minimum functionality that is necessary for BIOS option ROM execution and booting a BIOS operating system.

The CSM has the following key features:

- Allocates memory below 1 MB for legacy use
- Can be configured through the Setup utility
- Converts the UEFI memory map into E820 format
- Determines boot devices
- Initializes BDA, EBDA, and CMOS with the correct values
- Locates and loads the Compatibility16 image into memory
- Locates load and execute legacy PCI option ROMs
- Produces the necessary tables (MPS, PCIIRQ, APCI, SMBIOS)
- Provides a thunk interface for communication with Compatibility16
- Sets up hardware for legacy operating system use
- USB legacy SMM handler interface

Operating-system deployment and boot

There are two scenarios in which the CSM is involved in booting an operating system:

- The CSM supports installing and booting into a UEFI-compliant operating system when a supported hardware device is being controlled by a PCI option ROM through the UEFI thunk driver.
- The CSM supports installing and booting to supported BIOS operating systems.

UEFI-compliant operating systems

The CSM is responsible only for providing an environment in which a legacy option ROM can be installed and executed.

BIOS operating systems

The CSM is responsible for providing a complete operating environment that is compatible with what BIOS would provide.

Optimizing boot-time performance

The simplest way to achieve quicker boot times for a configuration is to install and boot UEFI-aware operating systems whenever possible. For deployments in which a UEFI-aware operating system is not available, this section introduces concepts and techniques for optimally configuring adapter support for legacy boots.

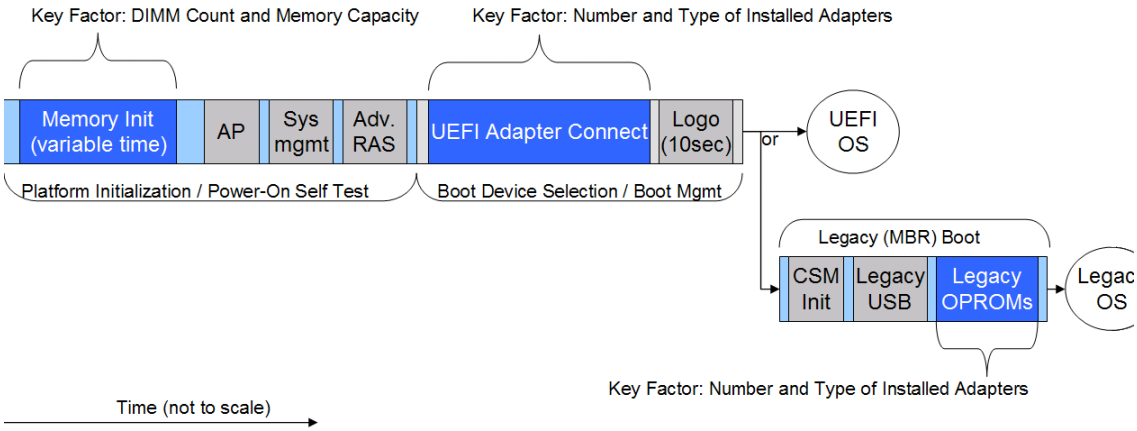
The major variable determinants of server boot-time performance are how much memory capacity is available and what adapters are installed. Other determinants of boot-time performance are inherent to the design and core technologies of the server design (such as CRTM/TPM, platform self-test, and power management) and are not configurable. Always use the latest available firmware, because any optimizations to these core features will be in the latest firmware releases.

Memory

The more memory that is installed, the more there is to initialize ECC and test. You can install less memory, but that usually does not result in significant boot-time improvement. The best approach for optimizing boot time and memory use is to balance DIMMs and memory capacity among installed processors. Balancing memory optimizes memory initialization on servers with integrated memory controllers (such as Intel Xeon® 5500 based servers). For details, see the *Optimizing the Performance of IBM System x and BladeCenter Servers using Intel Xeon 5500 Series Processors* white paper.

Adapters

Some classes of server adapters, such as network or RAID controllers, can take considerable time to initialize in the pre-operating-system UEFI or BIOS environment. Because IBM UEFI firmware simultaneously supports both UEFI and BIOS boot mechanisms, there can be unwanted repetition of adapter initialization when BIOS operating systems are booted. This repetition can occur with an adapter that includes native UEFI device drivers in addition to BIOS code on its adapter ROM. For suggested approaches to tuning adapter support for better boot-time performance, see “UEFI and BIOS adapter support details” and “Enabling and disabling adapter ROM support.”



Relative determinants of boot-time performance

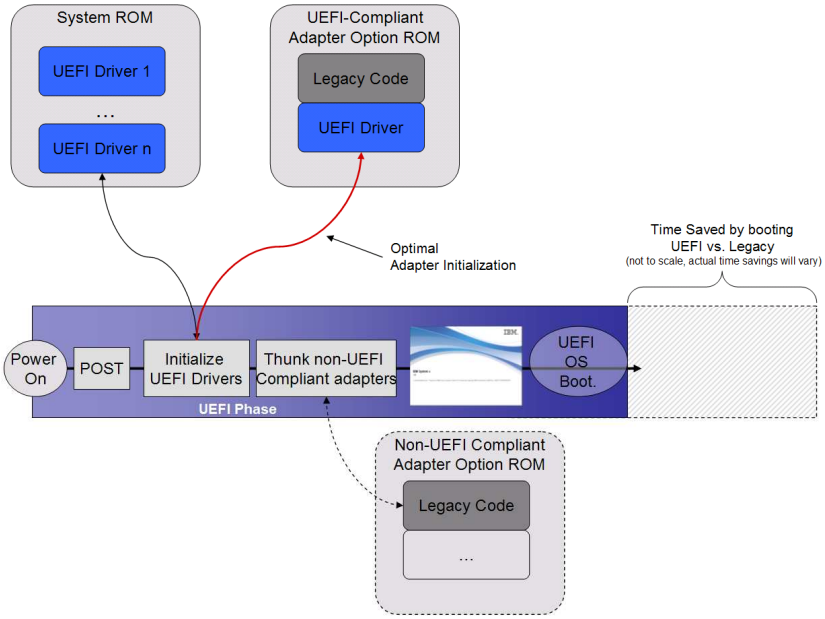
UEFI and BIOS adapter support details

There are three places where adapters can be initialized:

- **UEFI:** Driver execution environment (DXE) dispatcher and boot device selection (BDS) connect controllers.
- **UEFI legacy compatibility:** 16-bit thunk device drivers initialize legacy option ROMs and provide a UEFI compatibility wrapper.
- **Legacy compatibility module boot process:** Legacy option ROMs are initialized as part of the legacy boot process (only when booting legacy, non-UEFI-aware operating systems).

Optimal scenario

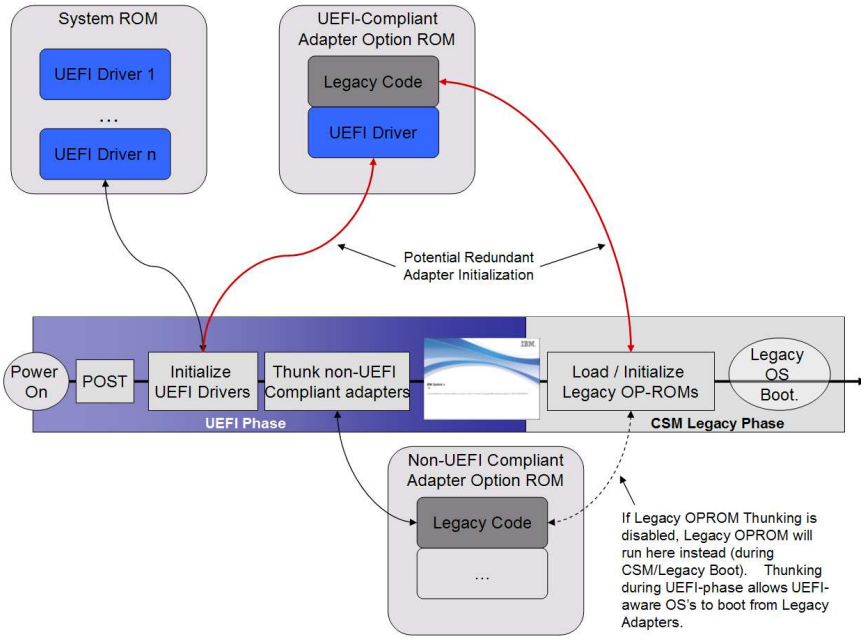
The best way to ensure the fastest boot time is to use UEFI-compliant adapters and a UEFI-aware operating system. The following figure illustrates the firmware interactions of an adapter with both a UEFI-compliant device driver and legacy ROM. The legacy code of the adapter is never invoked; therefore, the time penalty of that initialization is not incurred. Because the server is booting a UEFI-aware operating system, there is no need for BIOS compatibility for this boot.



Optimal scenario: Default dual BIOS/UEFI-compatibility adapter behavior during a UEFI boot

Non-optimized BIOS boot scenario

The following figure illustrates the worst-case scenario, in which an adapter with both a UEFI-compliant device driver and a legacy option ROM has its UEFI device driver connected during UEFI preboot and its legacy option ROM run during a BIOS boot. This redundancy is due to the UEFI preboot requirement to detect whether there is bootable media behind the adapter. The preferred method of doing this is to connect the UEFI device driver and check for bootable media. If the media contains a UEFI-aware operating system, there is no redundancy; however, if the media contains a legacy master boot record (MBR) operating system, the redundancy occurs because UEFI unloads (including the adapter UEFI device driver) and loads the compatibility support module (CSM), which calls the adapter legacy option ROM.

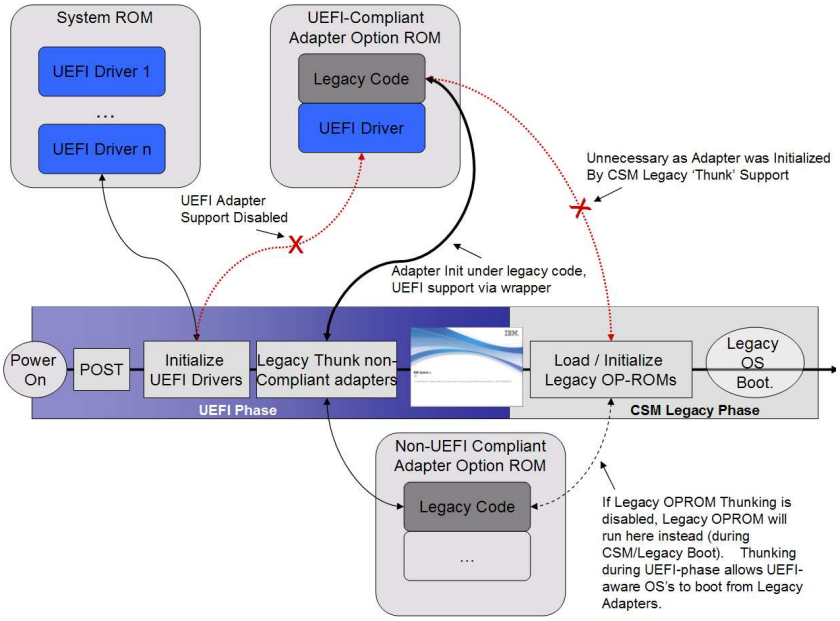


Worst-case scenario: Default full-compatibility adapter behavior during a BIOS boot

Optimized BIOS boot scenario

The following figure illustrates a scenario in which the server is going to boot a BIOS operating system and, therefore, you can optimize boot-time performance by disabling UEFI support for the adapter. The result depends on the PCI class of the adapter:

- **Storage adapters:** The IBM UEFI firmware invokes thunking support to provide a UEFI compatibility wrapper, enabling the UEFI boot manager to detect the media that the adapter is managing, if UEFI or legacy boot targets are available. This is critical, because if the UEFI boot manager cannot detect the media, it cannot recognize that an operating system is available to be booted.
- **Other adapters:** The UEFI boot device selection (BDS) logic cannot recognize that an operating system is available to be booted, so you must insert the **Legacy Only** flag above the category that the adapter is in (for example, **Network**). This flag instructs the firmware to assume that a BIOS operating system is available to be booted and that the compatibility support module (CSM) will detect it after the legacy adapter ROM is called.



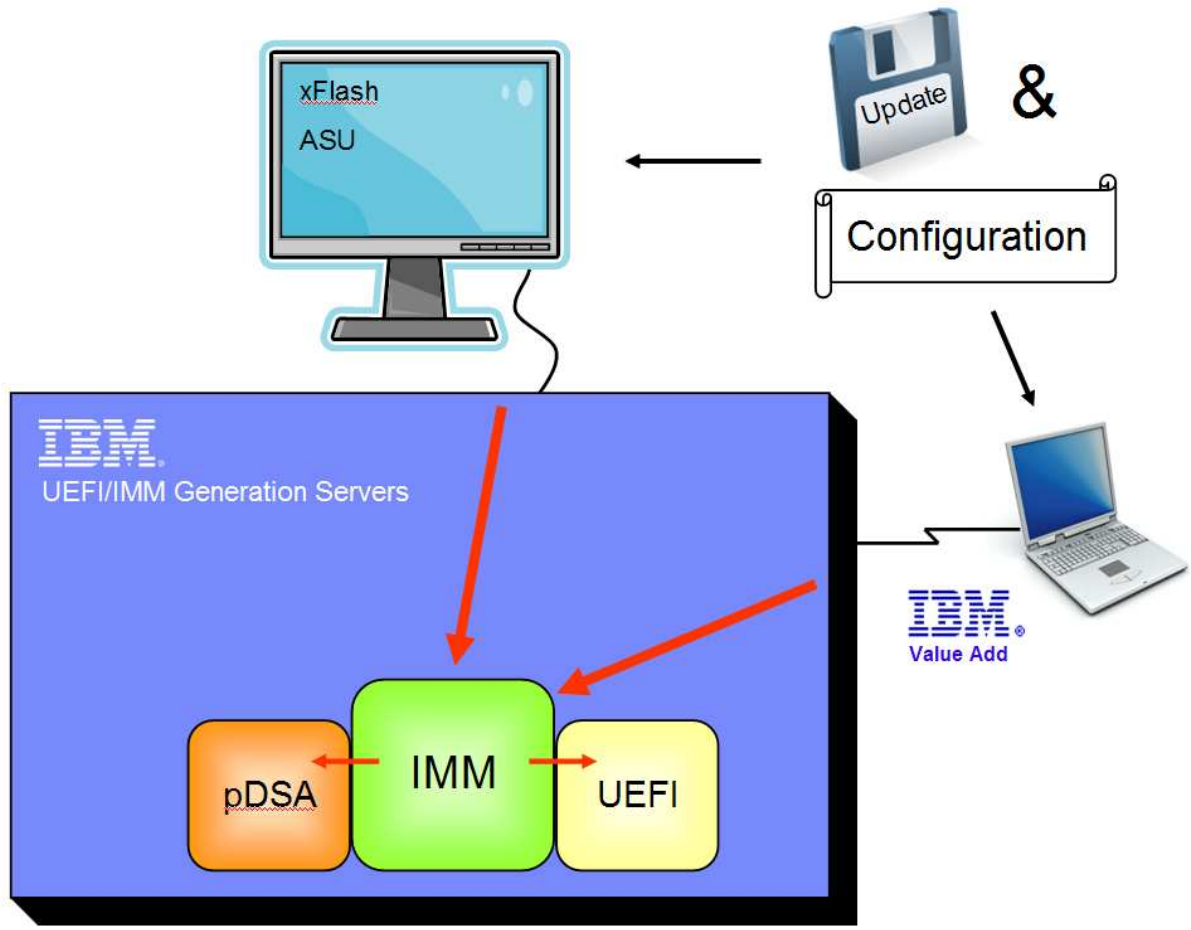
Optimized BIOS boot scenario: Adapter behavior with UEFI support disabled on UEFI/BIOS adapter ROM during a BIOS boot

Firmware configuration management

BIOS servers use battery-backed CMOS memory for storing firmware settings such as memory mode, speed selections, and server boot order. Problems with this design include space limitations and the fact that the settings require build-specific CMOS maps to be decoded (limited systems-management tool options) and are updatable only when the server is fully powered on and booted.

In contrast, UEFI System x servers use a two-tier, nonvolatile flash storage design for storage of server firmware settings. In the first tier, settings are stored in UEFI NVRAM variables, to which the UEFI code has direct and fast access. In the second tier, settings are modeled (XML), stored, and managed by the integrated management module in its NVRAM. This design allows for new capabilities while preserving the availability that is inherent to simpler, more functionally limited design.

In UEFI servers, settings can be managed, cloned, and restored while the server is powered off and connected to ac power. In BIOS servers, the Advanced Settings Utility (ASU) provides approximately 80% coverage of firmware settings; in UEFI servers, the ASU provides full coverage of the settings that are available in the Setup utility.



Adapter and add-on device configuration management

The method that you use to configure adapters depends on their compliance with UEFI and EFI.

Driver Configuration Protocol (DCP)

You can configure UEFI 2.0 and earlier EFI-compliant adapters by using the Setup utility (**System Settings → Adapter and UEFI Drivers**). To display all the driver instances that support DCP, press Enter when `refresh list` is displayed. To start the DCP user interface for a driver, highlight the driver and press Enter.

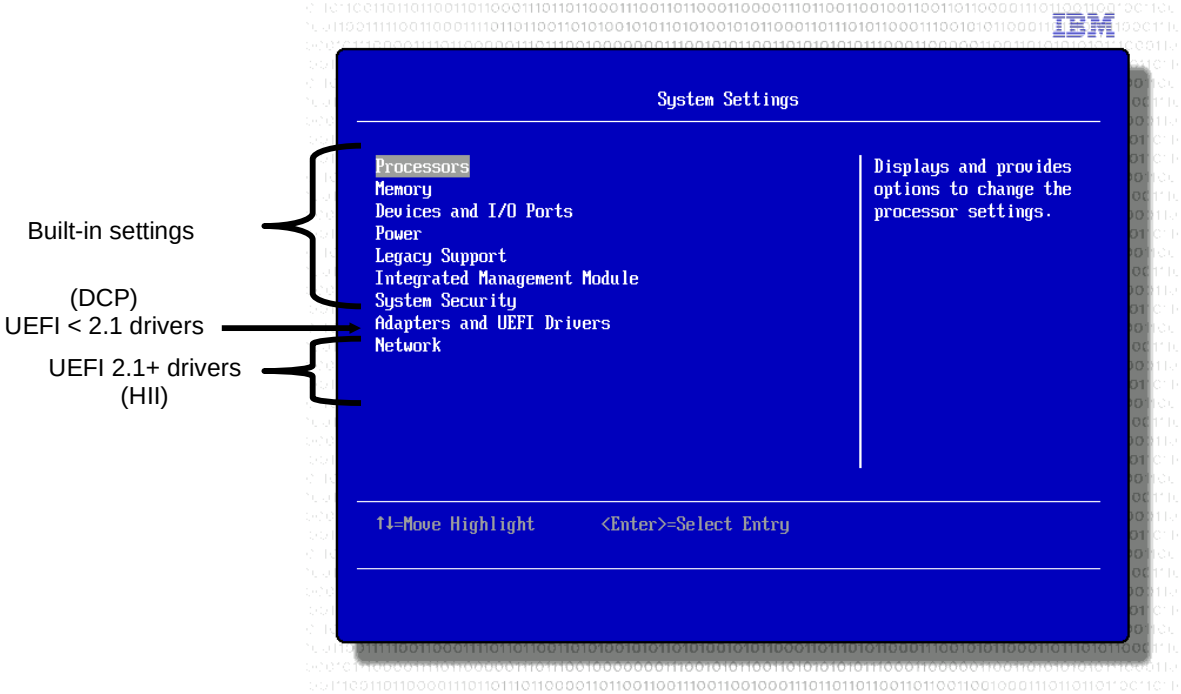
Hardware-Independent Imaging (HII) Configuration Access Protocol

UEFI 2.1 and earlier UEFI-compliant adapters are managed through the Setup utility (**System Settings → *device***). The HII configuration database contains the individual adapter settings grouped into forms, which enables you to add adapters without exiting from the Setup utility.

Legacy adapters

You can manage adapters that do not have UEFI or EFI device drivers by using escape sequences, such as Ctrl+A, during the POST/handoff process (as you would with BIOS-based servers).

Note: Some adapters can be managed through both the DCP and the HII Configuration Access Protocol. See the adapter documentation for information about which one to use.



Firmware Reliability, Serviceability, and Availability (RAS) features

IBM UEFI firmware on System x servers has the following RAS features:

Locked CRTM region:
Critical early platform initialization code is locked as part of the core root of trust for measurement (CRTM) region. This image is locked before CRTM-phase completion; therefore, only CRTM code can update the CRTM region. For this reason, after a new firmware update is applied, the server performs the CRTM update on the following boot (that is, the CRTM update is staged at the time of the flash update and is applied on a subsequent boot).
Note: The CRTM update process causes the server to perform a power-good reset during the CRTM-update POST cycle. This is not a fault.

IBM UEFI firmware volume integrity service:
This function runs shortly after the processors are started, and it checks the integrity of the firmware volumes that will used in later during POST, preboot, and run time.

Power sequence watchdog:
This is an integrated management module (IMM) timer that ensures that the IBM UEFI firmware has reached the point where it would have validated the firmware volume integrity. If the server fails to turn off this timer within the specified period, the IMM activates the automated BIOS recovery (ABR) policy. Failure to deactivate this timer is often the result of corrupted or missing UEFI code or inability of the processor to begin its fetch/decode/execute cycle.

POST watchdog:
This is an IMM timer that covers the entire power-on self-test period. The server deactivates this timer when POST is completed (at approximately the same time that the splash screen is displayed). If the server fails to turn this timer off within the specified period, the timer expires, and ABR is activated.
Note: This feature is disabled by default, but you can enable it through the Setup utility (**Settings → Integrated Management Module**).

Automated BIOS (UEFI) recovery (ABR):
System x servers have a dual-bank UEFI ROM. ABR is the policy engine for determining under which conditions the server automatically switches from the primary bank to the backup bank. ABR is able to detect the failure of the server to run POST (because of code corruption), according to inputs from CRTM, the firmware volume integrity service, and the IMM watchdog timers.

When one of the conditions for ABR is met (for example, when UEFI detects that a firmware volume that contains POST device drivers that it will need later is corrupted), the IMM switches the server from the primary bank to the backup bank. The server then boots from the code in the backup bank, using the same firmware settings as the primary bank uses.

When an ABR occurs, you can still use the server, although you should recover the primary bank as soon as production demands allow, to restore redundancy.

To enable the ABR functionality, you must keep the IBM UEFI firmware in the backup bank up to date. The online update utilities provide a switch that enables you to update the firmware in the backup bank.

To restore the primary UEFI firmware bank, complete the following steps:

1. Update the firmware in the primary bank, using a supported flash tool or the Web interface for the integrated management module or advanced management module.
2. Restart the server once while the backup bank is active. This ensures that if there was a corruption in the CRTM region, the backup CRTM can repair the image in the primary bank (using the CRTM-update capsule that was staged in step 1).

3. Use one of the following procedures to switch the server back to the primary bank:
   - Disconnect the server from ac power. Wait 15 seconds, and then reconnect the server to power.
   - Restart the server and press F3 when the splash screen is displayed.

Three consecutive boot failure (for UEFI firmware build dates prior to 2 February 2010): If the server is unable to complete the UEFI preboot process three times in a row, the server restores default settings and automatically starts the Setup utility, which you can use to correct possible configuration problems and reboot the server.

Note: You can use IPMI tools to remotely recover a server that is experiencing three boot failures:

- Use the Advanced Settings Utility (ASU) to update the configuration remotely.
- Use IPMI system power events (reset, power off then on) to restore the server to your current settings.

Three consecutive boot failure (for UEFI firmware build dates 2 February 2010 and later): If the server is unable to complete the UEFI preboot process three times in a row, the server temporarily uses the default settings and gives you an opportunity to start the Setup utility to correct possible configuration problems. If you do not press F1 to start the Setup utility, the server reboots with your current settings. If the server is still unable to successfully run POST after three attempts, the second three-boot failure be triggered, the server runs POST using default settings, and you are prompted to press F1 to start the Setup utility.

If the three consecutive boot failure recovery code is invoked three times in a row (for a total of nine boot attempts with your settings and three boot attempts with default settings), the server issues a fatal configuration error event and power off.

You can correct configuration errors remotely by using the ASU. If the server has already shut down after excessive boot attempts, you can power it on remotely by using IMM/AMM web, CLI, or IPMI power commands.

Note: Preboot hangs are typically associated newly added hardware, a recent configuration change, or repeated power interruptions.

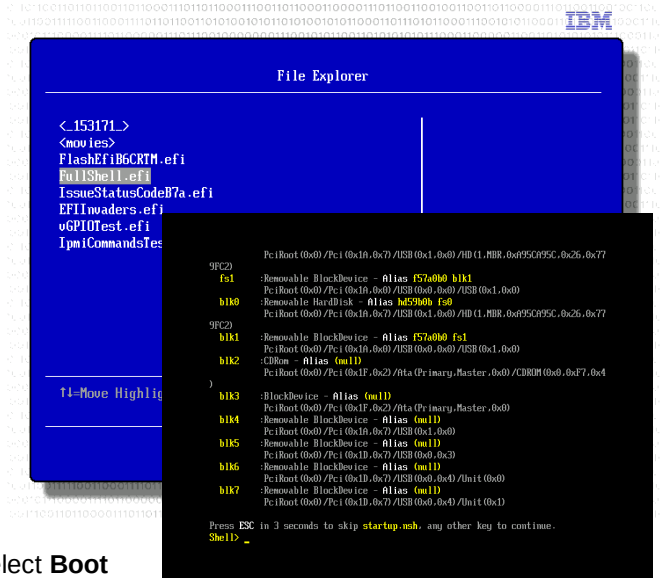
The UEFI shell

The UEFI shell is a standards-driven UEFI application that supports a command-line interface, a set of APIs, built-in commands, library support for external commands, and scripting capabilities. The shell is intended for advanced management and scripting of preboot functions.

Security consideration: The shell is an advanced environment with direct hardware access, including potential unrestricted access to available file system partitions. You might want to set a Password Authentication Protocol (PAP) password to prevent unauthorized users from starting the Setup utility and starting the UEFI shell or other potentially harmful UEFI applications.

Obtaining the UEFI shell

To obtain the UEFI shell, go to **http://www.tianocore.sourceforge.net** and click **EFI Dev Kit (EDK)**. From the official releases in **Docs and files**, download the latest EDK file. Extract the fullshell.efi file or the baseshell.efi file. The fullshell.efi file contains a more complete set of built-in commands.



Starting the shell and adding it to the boot order

To start the shell, start the Setup utility and select **Boot Manager → Boot From File**. Navigate to your USB key or other FAT-formatted media and select the shell.efi application.

To add the shell to the boot order list, start the Setup utility and select **Boot Manager → Add Boot Option** (or **Add WOL Boot Option**). Navigate to your USB key or other FAT-formatted media and select the shell.efi application.

Large-scale deployment considerations

Some adapters and devices require that the configuration be performed from a UEFI-shell-based utility (typically provided by the same vendor as the adapter). It can be unwieldy and inefficient to manually enter the shell and run the necessary commands and applications on every server in your enterprise. Instead, consider automating the process so that each server automatically boots to the shell and runs a shell script that automates the configuration steps. For more information, see “Configuring servers for automated UEFI deployments.”

UEFI event log design summary

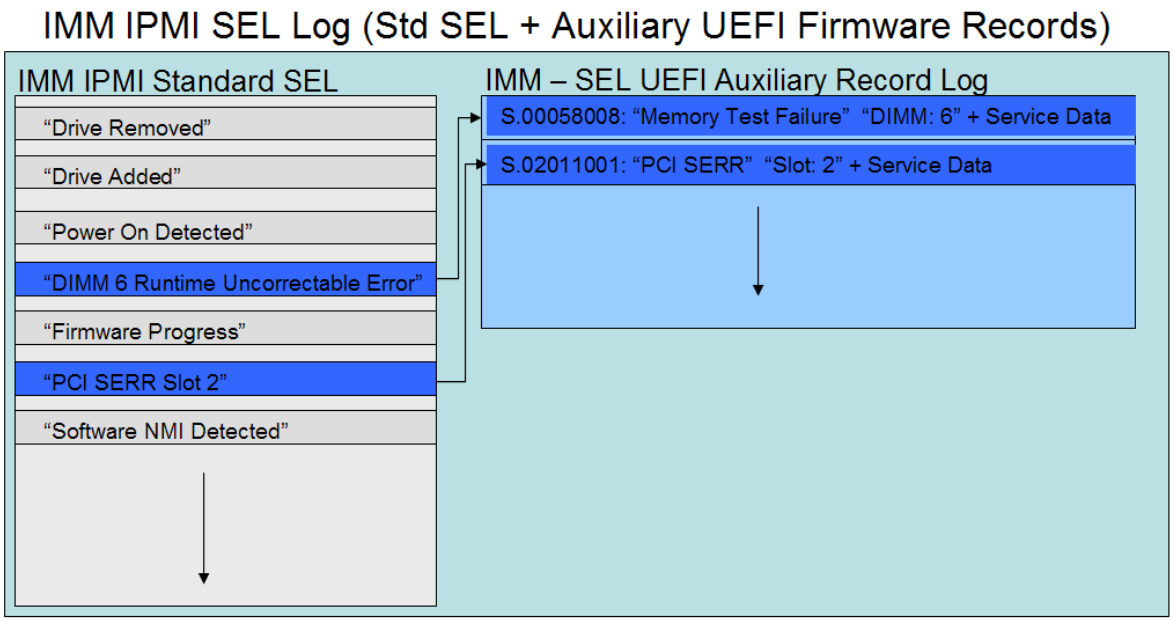
UEFI does not have a log of its own, but instead, its events are combined with the IPMI event log (also known as the system-event log, or SEL) that the integrated management module (IMM) maintains. UEFI diagnostic codes and service data are relayed to the IMM, which then generates the applicable standard IPMI event (fixed enumeration) and LED action and adds the UEFI diagnostic record and service data to a system-event-log-associated repository called the IPMI aux log.

Each UEFI event that is logged through the system-event log (all critical and warning events) produces both a standard entry (per the IPMI 2.1 specification) and a UEFI diagnostic code. The UEFI diagnostic code is associated with the standard entry and therefore shares the same timestamp and other log metadata. For example, a UEFI event is logged as follows:

```
12 PCI Fatal Error
 --> [S.2011001]
```

The first line of this event is the IPMI 2.1 standard entry. The second line is the UEFI diagnostic code; you can select this item to display the underlying firmware event.

UEFI events remain associated with their corresponding SEL entries until you clear the system-event log, at which point both the standard entry and the UEFI firmware diagnostic record (and service data) are cleared from the IPMI aux log.

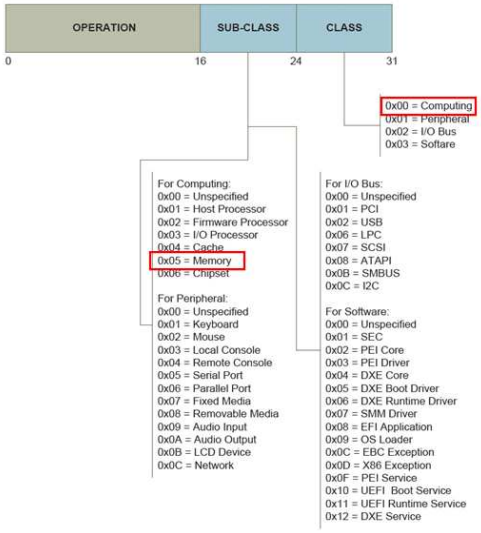


UEFI firmware events

Firmware events can occur in two phases of the server operational life cycle:

- **Preboot:** Hardware POST, adapter initialization, and operating-system handoff.
- **Run time:** For certain hardware errors, firmware events might be generated. Errors such as memory, PCI, chip set, or processor errors might cause firmware events to be issued.

Each firmware event has a 32-bit UEFI error code that is based on the class, subclass, and specific event ID. The format of a UEFI error code is as follows:



e.ccssddoo

e = Error severity
I = Information
W = Warning
S = Severe
cc = Class
ss = Subclass
dd = Defines whether the operation code is IBM-defined or from the UEFI Platform Initialization (PI) 1.2 standard
00 = PI industry-standard event definition
80 = IBM-unique event definition
oo = Operation

For example, the error code S.00058008 indicates that a DIMM failed the memory test and is decoded as follows, according to the PI 1.2 specification:

Severity = Severe (S)
Class = Computing (00)
Subclass = Memory (05)
Event definition = IBM-unique (80)
Operation = DIMM failed the memory test (08)

A firmware event comprises four primary entities:

- **Event type:** Describes the event as Progress, Error, or Debug (not used in production builds)
- **Event value:** Uniquely identifies the event through a three-level hierarchy of class, subclass, and operation
- **Instance:** Identifies the hardware entity that the event pertains to or, for software events, a reason code
- **Service data:** A chip set or other detailed error capture data that IBM service or product engineering can use for failure analysis and root-cause investigations

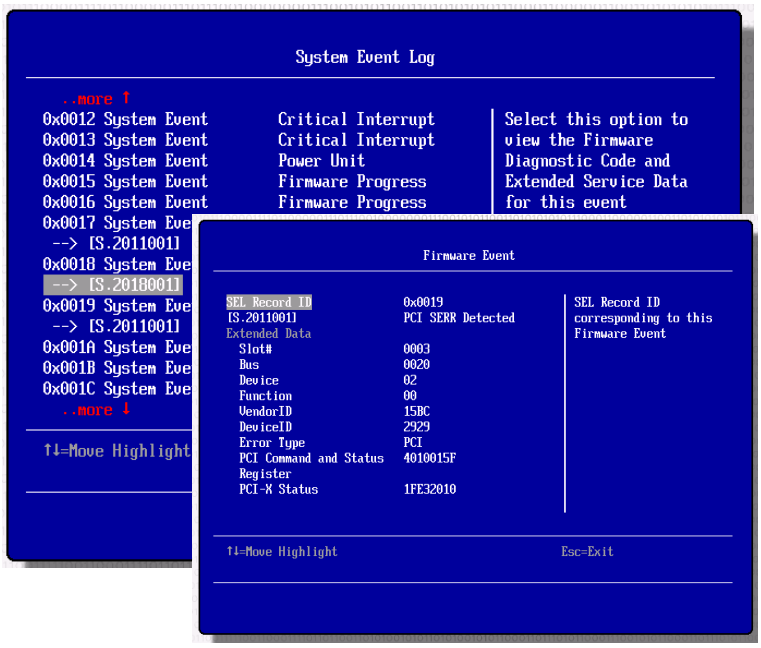
Drilling down to firmware events

Many firmware events are mapped to IPMI platform events and are associated at the time of issuance with the IPMI event log. This association enables you and IBM service to drill down into IPMI events to determine the underlying firmware event. This is useful in two ways: several firmware events can be mapped to a single IPMI event, and firmware event records usually include extended service data (such as register states and additional information).

To view firmware events, start the Setup utility and select **System Event Logs**. Then, use either of the following choices:

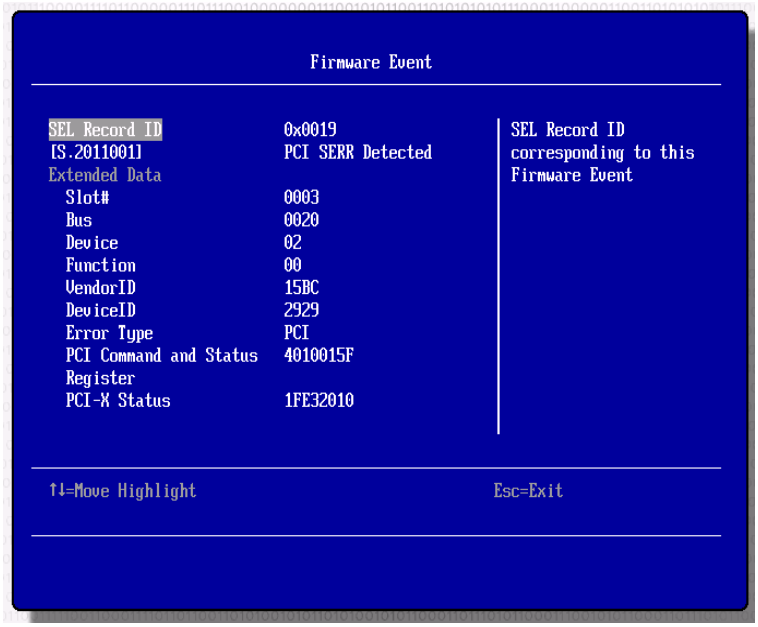
- Select **POST Event Viewer** to display only the events that have occurred during the current boot.
- Select **System Event Log** to display the IPMI event log as it would be shown in the integrated management module or advanced management module web interface, with one important addition: you can drill down into an IPMI event that was triggered by a firmware event to see the firmware event. The firmware event data includes the severity, diagnostic code, and any extended service data that is available.

In the system-event log, each UEFI event that was triggered by a firmware event is designated by an arrow (-->) followed by a UEFI diagnostic code. To view the details of the underlying firmware event, highlight the UEFI diagnostic code and press Enter.



Runtime PCI firmware event records

The IBM UEFI firmware handles PCI errors, and when they are unrecoverable, they are reported to the system-event log, and an associated entry is added to the firmware service log.



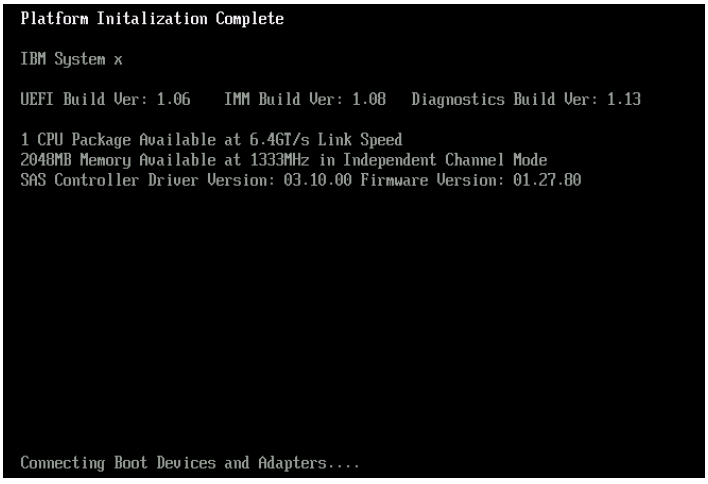
IPMI restricts what can be encoded in the standard PERR/SERR (PCI-X) and uncorrectable bus (PCIe) standard events. An IPMI event indicates the error type and the slot number, if the error involves an adapter. If the IPMI event indicates slot number 0, the error probably involves a device that is integrated on the system board.

If an IPMI event indicates a PCI error but no slot number is indicated, you can drill down to the firmware event to view additional information about the error, including the physical and logical location information (slot, bus, device, and function numbers), the adapter or device vendor ID and device ID, and the PCI failure data (status and command registers).

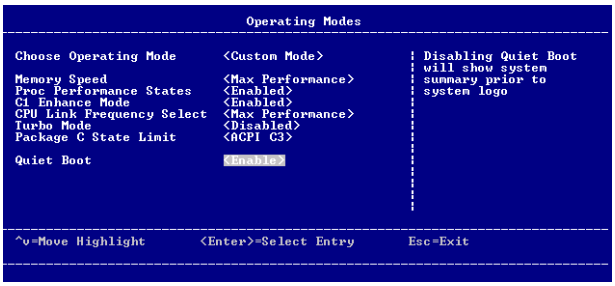
Note: If unrecoverable PCI errors occur after a device-driver or adapter firmware update and the firmware event record indicates that one of the devices that you updated (for example, the check vendor/device ID) is causing the errors, consider returning the device driver or firmware to the previous level and contacting IBM service or the adapter vendor for assistance.

Displaying the system summary via remote console or disabled Quiet Boot

When platform initialization (POST) is complete, the server goes into the boot device selection (BDS) phase, at which point input/output devices are connected as well as adapters that might be used as part of the boot process. During this time, if Quiet Boot is enabled, a logo splash screen is displayed on the primary display (VGA); if Quiet Boot is disabled, a text-based summary of the system hardware and firmware entities is displayed.



To enable or disable Quiet Boot in the Setup utility, click **System Settings → Operating Modes → Quiet Boot**.



To enable or disable Quiet Boot in the Advanced Settings Utility, use one of the following commands:

```
asu set uEFI.QuietBoot enable
asu set uEFI.QuietBoot disable
```

Regardless of the Quiet Boot setting, the system summary is always displayed on text-based remote consoles, including Serial over LAN.

Disabling adapters and integrated devices

You can disable adapters and integrated devices that are not needed for any phase of the system life cycle (POST, boot, or operating-system run time). Disabled devices are not visible to the operating system. Disabling unneeded boot devices might improve boot times, in some cases significantly.

To disable a device, in the Setup utility, select **System Settings → Devices and IO Ports → Enable / Disable Onboard Device(s)**.

Enabling and disabling adapter ROM support

Adapter support falls under two broad categories: UEFI and legacy. IBM UEFI firmware includes legacy adapter compatibility thunk support, which can provide a UEFI wrapper for legacy option ROMs, enabling an adapter to be used in a UEFI context for booting.

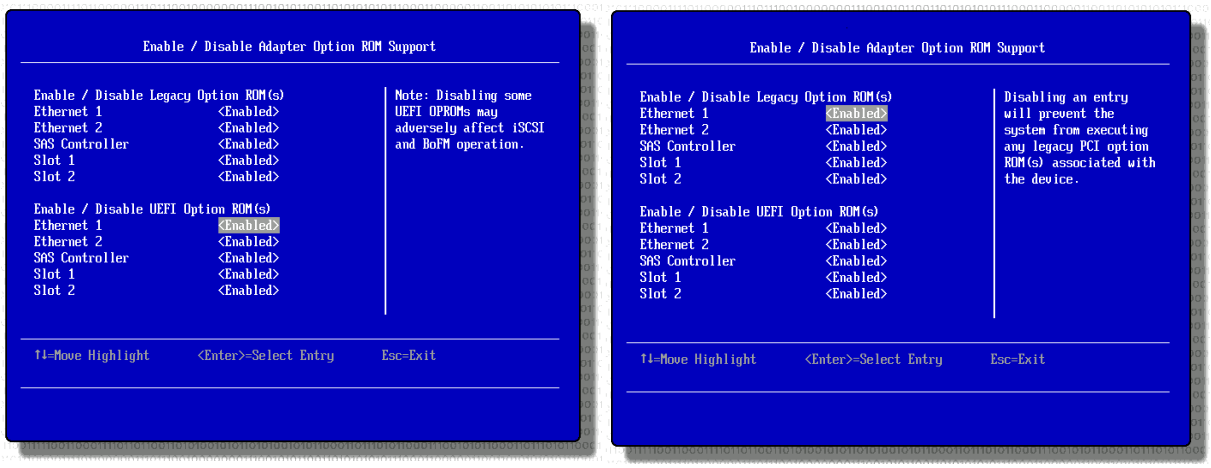
Legacy adapter support is required in the following scenarios:

- To boot a legacy operating system from an adapter (over the network or locally)
- To boot a UEFI-aware operating system from a non-UEFI-compliant adapter

UEFI adapter support is required in the following scenarios:

- To boot a UEFI operating system from the adapter when legacy support has been disabled or when thunk support does not provide a UEFI-compatibility wrapper (for example, when the adapter is of a nonstandard PCI device class)
- For boot-neutral functions such as BladeCenter Open Fabric Manager and initial iSCSI support.

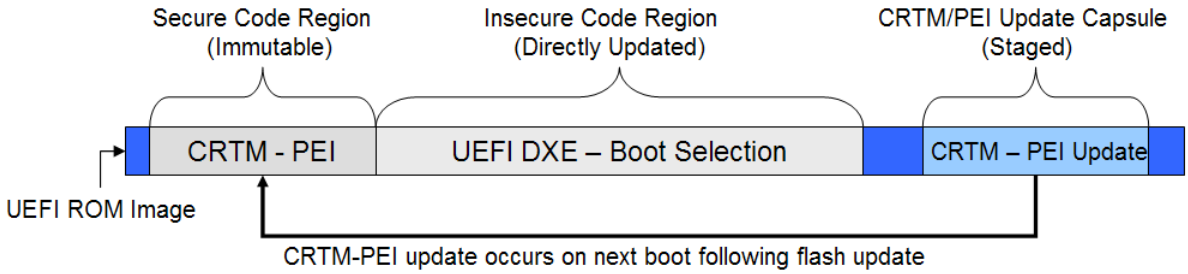
To disable or enable adapter ROM support, in the Setup utility, select **System Settings → Devices and IO Ports → Enable / Disable Adapter Option ROM Support**. The first list contains the adapters and slots for legacy support, and the second list contains adapters and slots for UEFI support.



Recovering from UEFI firmware corruption

The UEFI firmware is composed of two distinct regions:

- **Secure code (locked-immutable) region:**
  1. The flash update process stages the new trusted-code update capsule in a staging area of the UEFI ROM.
  2. During the first boot after the firmware is updated, the active trusted code executes up to the point at which it is able to detect the new trusted-code update capsule.
  3. The active trusted code verifies that the trusted-code update capsule has the appropriate signing key.
  4. If the key is valid, the active trusted code copies the trusted-code update into the trusted-code region of the ROM.
  5. The server restarts to complete the update and resume normal operation.
- **Insecure nontrusted code (unlocked-updatable) region:**
  Code in this region is directly updated by the flash update process and is validated by the IMM flash manager (communicated via the flash tool or web interface).



How to determine whether recovery is necessary

You must recover the firmware if the server is unable to complete POST (that is, the server firmware splash screen and the <F1> Setup prompt are not displayed), the three-boot failure feature is unable to recover the system, and either of the following conditions occurs:

- The system-event log indicates firmware corruption (BIOS).
- The server is running from the backup flash memory bank because of Automated Bank Recovery (ABR).

Note: If the server has only non-POST symptoms but no corruption events are logged in the IMM event log and no fault or board LEDs are lit, remove any recently added hardware and restart the server four times to ensure that the three-boot failure feature is invoked. If you are still unable to recover the system, follow the UEFI firmware recovery procedure.

In-band UEFI firmware recovery

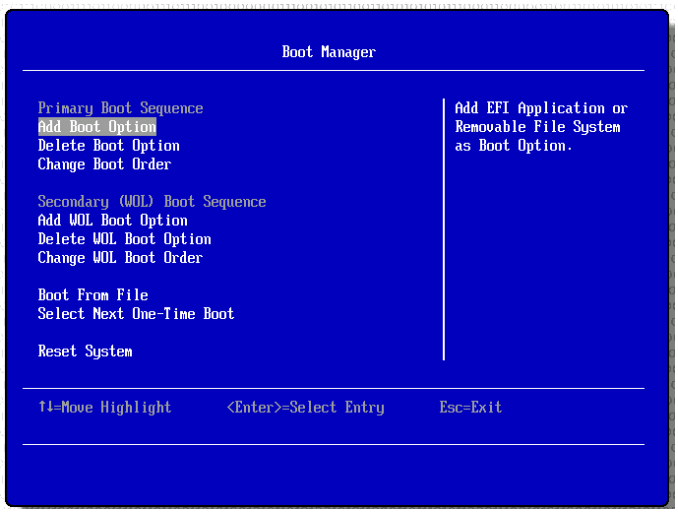
This procedure assumes that the primary flash memory bank is the affected or corrupted bank. If the backup bank is affected, the procedure is similar. To recover the UEFI firmware, complete the following steps:

1. Verify that the server is running from the backup bank. If Automated Bank Recovery (ABR) was invoked, server operation switches to the backup bank automatically. If the server is running from the primary bank, switch server operation to the backup bank (see the server *Problem Determination and Service Guide* for instructions).
2. Boot the server to the operating system or bootable update media, and update the primary bank to the last known good firmware level. The server might power-cycle itself during this boot, which might indicate that the trusted-code module in the backup bank detected a pending update for the primary bank and performed that update (recovering from possible corruption of the code in the primary bank).
3. When the server completes POST from the backup bank, return server operation to the primary bank:
   - If the server operation was automatically switched to the backup bank through ABR, power-cycle or restart the server and press F3 at the server firmware splash screen to restore server operation to the primary bank.
   - If you manually switched server operation to the backup bank, follow the instructions in the Problem Determination and Service Guide to restore server operation to the primary bank.
4. Verify that the server successfully completes POST from the primary bank. If the server does not successfully complete POST, replace the system board.

Using the Legacy Only flag

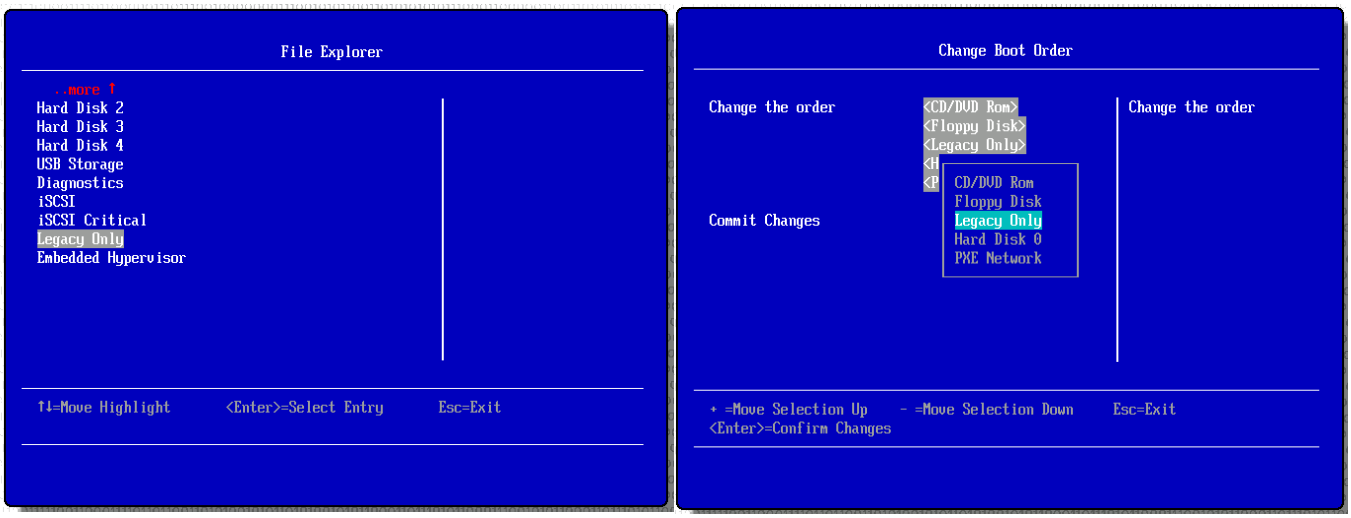
There are two scenarios in which the **Legacy Only** flag is required:

- You want to boot the BIOS target on dual-boot media. For example, if you want to install Windows Server 2008 as a BIOS installation instead of a UEFI installation from a dual-boot installation DVD, you must place the **Legacy Only** flag above the DVD/CD boot target. This instructs the firmware to boot the BIOS target, even though a UEFI target is available.
- The BIOS-operating-system deployment is on media that is not available to the IBM System x Server Firmware (that is, the media is managed by a non-UEFI-compliant storage controller on which UEFI support cannot be emulated through thunking). The **Legacy Only** flag above the boot targets instructs the firmware to attempt a BIOS boot, even though it cannot detect the operating-system target.



To add the **Legacy Only** flag to the boot order list, complete the following steps:

1. Restart the server and press F1 to start the Setup utility.
2. Select **Boot Manager**.
3. Select **Add Boot Option** or **Add WOL Boot Option**.
4. Select **Legacy Only** and press Enter. The **Legacy Only** flag is added to the end of the boot order list.
5. Select **Change Boot Order** or **Change WOL Boot Order** and press Enter.
6. Highlight **Legacy Only** and press + until the **Legacy Only** flag is above the first boot target to which the **Legacy Only** flag applies. Press Enter.
7. Select **Commit Changes** and press Enter.
8. Exit from the Setup utility.



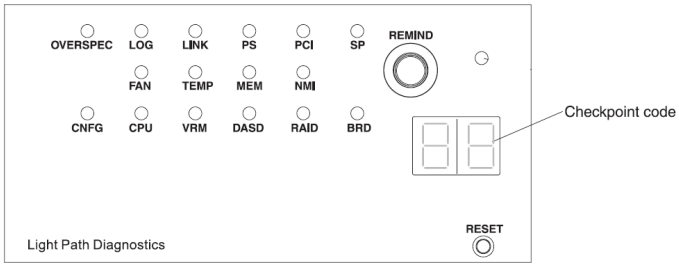
UEFI checkpoint codes

Note: This section does not pertain to System x3850 X5 servers. See the System x3850 X5 *Problem Determination and Service Guide* for details about checkpoint codes for that server.

UEFI checkpoints are 8-bit hex codes that represent server progress and, rarely, might reflect firmware or hardware errors early in POST (the power-on self-test). The checkpoints themselves are not intended to be used as deterministic indications of fault or progress, because different entities use the underlying I/O port that is mapped to the checkpoint code display (in servers that have a light path diagnostics panel).

Rarely, IBM service might request checkpoint values to assist in problem determination. On their own, checkpoints provide very little value to the user. The user should use the light path diagnostics LEDs, firmware events, and the IMM event log as indications of fault conditions.

General UEFI checkpoint ranges:
0x00-0x0F Hardware/CPU init code
0x10-0x1F PEI phase device drivers
0x20-0x9E DXE phase device drivers
0xA0-0xAF QPI init status
0xB0-0xBF Memory init status
0xC0-0xFF Miscellaneous discrete events



Select specific error checkpoints:
0xE0-E5 Processor/bus initialization failure
0xE8 No memory detected or available (confirm the MEM and CFG LEDs also)
0xED Invalid mixed DIMM configuration
0xEE Invalid DIMM population
0x9F POST complete

Note: Use caution when you interpret checkpoint values. Because multiple hardware, firmware, and adapters can issue checkpoints, the issuing entity might not be obvious. For example, for a memory event, the checkpoint data should be correlated with a lit MEM LED on the light path diagnostics panel and one or more memory events in the system-event log.

For More Information

IBM System x and xSeries Servers	**ibm.com**/systems/x/
IBM Rack Configurator	**ibm.com**/systems/x/hardware/configtools.html
IBM Configuration and Options Guide	**ibm.com**/systems/xbc/cog/

Legal Information

IBM Systems and Technology Group
Dept. U2SA
3039 Cornwallis Road
Research Triangle Park, NC 27709

Produced in the USA
January 2011

For a copy of applicable product warranties, write to: Warranty Information, P.O. Box 12195, RTP, NC 27709, Attn: Dept. JDJA/B203. IBM makes no representation or warranty regarding third-party products or services including those designated as ServerProven or ClusterProven. Telephone support may be subject to additional charges. For onsite labor, IBM will attempt to diagnose and resolve the problem remotely before sending a technician.

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at http://**ibm.com**/legal/copytrade.shtml.

Intel, Intel Xeon, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

IBM reserves the right to change specifications or other product information without notice. References in this publication to IBM products or services do not imply that IBM intends to make them available in all countries in which IBM operates. IBM PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

This publication may contain links to third party sites that are not under the control of or maintained by IBM. Access to any such third party site is at the user's own risk and IBM is not responsible for the accuracy or reliability of any information, data, opinions, advice or statements made on these sites. IBM provides these links merely as a convenience and the inclusion of such links does not imply an endorsement.

Information in this presentation concerning non-IBM products was obtained from the suppliers of these products, published announcement material or other publicly available sources. IBM has not tested these products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

MB, GB and TB = 1,000,000, 1,000,000,000 and 1,000,000,000,000 bytes, respectively, when referring to storage capacity. Accessible capacity is less; up to 3 GB is used in service partition. Actual storage capacity will vary based upon many factors and may be less than stated.

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will depend on considerations such as the amount of multiprogramming in the user’s job stream, the I/O configuration, the storage configuration and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

Maximum internal hard disk and memory capacities may require the replacement of any standard hard drives and/or memory and the population of all hard disk bays and memory slots with the largest currently supported drives available. When referring to variable speed CD-ROMs, CD-Rs, CD-RWs and DVDs, actual playback speed will vary and is often less than the maximum possible.