



# Intel<sup>®</sup> C Compiler for EFI Byte Code

## In-Depth

## Contents

Intel® C Compiler for EFI Byte Code.....	3
--	---

## Intel® C Compiler for EFI Byte Code

The Intel® C Compiler for EFI Byte Code enables the development of EFI Byte Code (EBC) images, allowing binary portability among multiple architectures. EBC images can be executed on EFI 1.10, UEFI 2.0 (or later) compliant systems, and can be executed on IA-32, Intel® 64, or Intel® Itanium® architectures. The Intel C Compiler for EBC delivers leading-edge size performance, offers source code compatibility, and comes with one year of customer support direct from Intel.

All features are for IA-32, Intel 64, and Itanium architecture-based systems unless otherwise noted).

- The compiler provides leading-edge, compact code-size performance.
- EBC images are binary compatible across IA-32, Intel 64, and Itanium architectures.
- The same EFI source code can be re-compiled with the EBC compiler. Developers can use a single source to create images for different target architectures including IA-32, Intel 64, or Itanium architecture-based processors.
- Since an EFI Byte Code (EBC) image can execute on IA-32, Intel 64, or Itanium architectures, a significant reduction in code-size space is realized, resulting in cost savings to card vendors. Another cost-saving feature enables card vendors to offer a single card to serve multiple market segments. The Intel C Compiler for EFI Byte Code creates an image in EFI Byte Code (EBC). This image can be executed by systems that implement the EFI 1.10, UEFI 2.0, or later specifications. These systems include an EBC interpreter that loads and interprets the EBC image, allowing an EBC image to be executed on multiple platforms and architectures including the IA-32, Intel 64, or Itanium architectures.

Typically, an EBC image is programmed into a PCI card's option ROM. Since an EBC image can execute on multiple architectures, a significant reduction of code-size space is realized. Another cost-saving feature is enabling card vendors to have one card that serves multiple markets.

Prior to using this compiler, it is recommended that a working native EFI image should already be developed. The EFI Development Kit (EDK) from <http://EDK.tianocore.org> is the preferred development environment for developing EFI compliant drivers. The EDK contains sample drivers and required reference code.

Features	Benefits
Compact code size	The compiler provides leading-edge, compact code-size performance.
Cross-architecture compatibility	EBC images are binary compatible across IA-32, Intel 64, and Itanium architectures.
Source compatible with multiple EFI drivers	The same EFI source code can be re-compiled with the EBC compiler. Developers can use a single source to create images for different target architectures including IA-32, Intel 64, or Itanium architectures.
Card vendors can save code-size space and use a single card to serve multiple market segments	Since an EFI Byte Code (EBC) image can execute on IA-32, Intel 64, or Itanium architectures, a significant reduction in code-size space is realized, resulting in cost savings to card vendors. Another cost-saving feature enables card vendors to offer a single card to serve multiple market segments.

## Optimization Notice

Intel® compilers, associated libraries and associated development tools may include or utilize options that optimize for instruction sets that are available in both Intel® and non-Intel microprocessors (for example SIMD instruction sets), but do not optimize equally for non-Intel microprocessors. In addition, certain compiler options for Intel compilers, including some that are not specific to Intel micro-architecture, are reserved for Intel microprocessors. For a detailed description of Intel compiler options, including the instruction sets and specific microprocessors they implicate, please refer to the “Intel® Compiler User and Reference Guides” under “Compiler Options.” Many library routines that are part of Intel® compiler products are more highly optimized for Intel microprocessors than for other microprocessors. While the compilers and libraries in Intel® compiler products offer optimizations for both Intel and Intel-compatible microprocessors, depending on the options you select, your code and other factors, you likely will get extra performance on Intel microprocessors.

Intel® compilers, associated libraries and associated development tools may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include Intel® Streaming SIMD Extensions 2 (Intel® SSE2), Intel® Streaming SIMD Extensions 3 (Intel® SSE3), and Supplemental Streaming SIMD Extensions 3 (Intel® SSSE3) instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors.

While Intel believes our compilers and libraries are excellent choices to assist in obtaining the best performance on Intel® and non-Intel microprocessors, Intel recommends that you evaluate other compilers and libraries to determine which best meet your requirements. We hope to win your business by striving to offer the best performance of any compiler or library; please let us know if you find we do not.

Notice revision #20101101

